MODELS OF GRAPH CLASSES AND APPLICATIONS

By

Zlatko Joveski

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

September 30, 2020

Nashville, Tennessee

Approved:

Jeremy P. Spinrad, Ph.D.

Akos Ledeczi, Ph.D.

Julie L. Johnson, Ph.D.

Mark N. Ellingham, Ph.D.

To Mihaela

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

Definitions

In this chapter, we provide a brief overview of basic concepts and terms used throughout the dissertation. Additional standard graph-theoretic notions not included here can be found in many graph theory and graph algorithms textbooks, including [12, 15, 104].

We primarily deal with finite undirected simple graphs and refer to them simply as graphs. Given a graph G, we denote its vertices as $V(G)$ and its edges by $E(G)$. We also denote the number of vertices in G by $n(G)$ and the number of edges by $m(G)$. When G is unambiguous, we denote the above notions as V, E, n, and m, respectively and we write $G = (V,E)$. We will use uv to denote an edge between vertices u and v.

The complement $\overline{G} = (V,\overline{E})$ of a graph $G = (V,E)$ is defined by $\overline{E} = \{uv \mid u,v \in V, u \neq v$, and $uv \notin E\}$. In some instances, we will also use the notation G^C for the complement of G. We say that a graph $G' = (V',E')$ is an *induced subgraph* of a graph $G = (V,E)$ if $V' \subseteq V$ and $E' = \{uv \mid uv \in E$ and $u,v \in V'\}$. We also say that G' is induced by V' and write $G' = G[V']$ or, if $V' = \{v_1, v_2, \ldots, v_k\}$, we sometimes write $G' = G[v_1, v_2, \ldots, v_k]$. Since we will only be dealing with induced subgraphs in this paper, we will often omit the word “induced".

We say that $V' \subseteq V$ is an *independent set* in G (or *empty subgraph* of G) if for all $u,v \in V'$, $uv \notin E$. We say that $V' \subseteq V$ is a *clique* in G (or *complete subgraph* of G) if for all $u,v \in V'$ such that $u \neq v$, $uv \in E$. An independent set (clique) S is *maximal* if there is no independent set (clique) S' in G such that $S \subsetneq S'$. Further, S is *maximum* if $|S|$ is the maximum possible size of an independent set (clique) in G. We will refer to the problem of finding a maximum independent set (clique) as INDEPENDENT SET (CLIQUE).

Given a vertex v of a graph G, we use $N(v)$ to denote the *open neighborhood* of v consisting of vertices adjacent to v in G. We use $N[v] = \{v\} \cup N(v)$ to denote the *closed neighborhood* of v. We call v *simplicial* if its closed neighborhood $N[v]$ induces a clique in G. The degree $d(v)$ of v is the number of edges in G incident with v. For simple undirected graphs we have $d(v) = |N(v)|$. We call vertices of degree 1 *pendant*.

We call two vertices $u,v \in V(G)$ *twins* if they share all their neighbors, except possibly them-

selves, i.e. when $N(u) - \{v\} = N(v) - \{u\}$. Further, we call two twins u and v *true* when they are adjacent and *false* when they are not.

Throughout this dissertation, we will denote by P_k and C_k the chordless path and chordless cycle on k vertices, respectively. We refer to the problems of finding a maximum P_k and a maximum C_k as the LONGEST CHORDLESS CYCLE and LONGEST CHORDLESS PATH problems.

We say that a graph G is *k-colorable* if each vertex in G can be assigned a color in the range $\{1,2,\ldots,k\}$ so that no two adjacent vertices are assigned the same color. We use COLORING to refer to the problem of finding the lowest possible value of k for which an input graph G is k-colorable.

Given an algorithm A that solves a particular problem, we will say that A is *efficient* if its running time is polynomial in the size of the input (e.g. if the running time of A is bounded from above by a polynomial in n and m, when the input is a graph G).

In this dissertation, we will primarily deal with problems on classes of graphs having a particular characterization or model. We will distinguish between three different types of graph algorithms on special classes of graphs: *model-based*, *promise*, and *robust* [95].

Model-based algorithms will require that the input graph is given together with its special model (not just its adjacency list or adjacency matrix representation). In *promise* and *robust* algorithms, the input graph will be given without an additional model, usually through its adjacency list representation. A *promise* algorithm operates on the basis of a "guarantee" that the input graph is a member of the restricted class of graphs that the algorithm is designed for. If this guarantee is not met, the algorithm need not function correctly or even terminate. A *robust* algorithm A, on the other hand, is required to produce a "correct" output in polynomial time, for any input. For an input graph that does come from the restricted class of graphs, A needs to solve the original problem in polynomial time. For an input graph G not from the class, A needs to either solve the original problem in polynomial time, or produce, in polynomial time, a certificate that G is not a member of the special class of graphs.

CHAPTER II

Introduction

Graphs, as a tool for modeling entities and relationships between them, are ubiquitous. For example, graphs are used to represent and study communication and interaction in offline and online social networks of individuals [78, 80, 90]. Further, Bayesian networks – which are modeled as directed acyclic graphs – are a fundamental tool in the theory of causal inference [31, 93, 110]. Many biological systems can be represented as graphs, including metabolic pathways, protein-protein interaction networks, and food webs [90, 106]. Graphs also play a very important role in large-scale information processing systems where structured and unstructured data is integrated into knowledge graphs on which further reasoning and inference is performed [94, 99].

In many application domains, the corresponding graphs satisfy special properties such as parameter (in)equalities, absence of certain forbidden subgraphs, having geometric intersection or containment models, or having certain types of vertex or edge orderings [15]. This can lead to restricted graph classes on which important optimization problems such as CLIQUE and INDEPENDENT set, that are NP-hard on general graphs, become tractable. In such situations, the design of efficient algorithms can take one of several forms. In some instances, we may be given a special model of the input graph, in addition to the the usual adjacency matrix or list representation, which facilitates the design of a model-based polynomial algorithm. It may also be possible to recognize that a graph belongs in the restricted class and reconstruct its special model in polynomial time. In other cases, the special property that graphs in the restricted class satisfy may be such that a robust polynomial algorithm is possible, without needing to be given, or to reconstruct, a model.

When solving optimization problems on graphs such as CLIQUE and INDEPENDENT SET, we are interested in finding graph substructures of a certain kind. However, identifying special substructures of graphs is not always of primary interest. Another important line of research is the study of how different processes evolve or spread over a graph as time passes, including information diffusion [75] and disease spread in social networks [89]. Decentralized coordination between entities positioned as nodes on a network can also be seen as a process over a graph. Different forms of decentralized coordination have been studied, including consensus [69, 73, 74], coloring

[69, 85], and bargaining [18]. An important question in these settings is to what extent properties of the underlying graphs, which may be coming from different restricted classes, affect the dynamics of a process [11, 90].

This dissertation is centered around models of graph classes. We explore how models can be used to define new graph classes that either generalize or combine multiple properties. Further, we study when and how such properties can help us design efficient algorithms for recognition and optimization problems on the class. We also consider how properties of graph class models can affect the dynamics of processes taking place over a graph. Our work consists of three main parts.

In Chapter III, we consider graph classes characterized by geometric intersection models. In particular, we introduce a novel model that generalizes the geometric intersection models of interval and permutation graphs. As a result, we obtain two new classes of graphs – simple IP-SEG and IP-SEG – that generalize interval and permutation graphs, but are not contained within the class of perfect graphs. We show that even though simple IP-SEG and IP-SEG graphs may contain large chordless cycles, there are limitations in how such cycles may be represented in the corresponding models. This allows us to identify several families of forbidden subgraphs for the classes. Leveraging properties of the geometric intersection model, we design polynomial model-based algorithms for the CLIQUE, INDEPENDENT SET, and LONGEST CHORDLESS CYCLE problems on IP-SEG graphs. We make initial progress on the recognition problem by presenting a polynomial recognition algorithm for the subclass of IP-SEG trees. We also discuss possible generalizations of the geometric intersection model of IP-SEG graphs that may preserve properties that allow the design of polynomial model-based algorithms for the above optimization problems.

In Chapter IV, we consider a different model used for characterizing or representing special classes of graphs – vertex orderings. We explore how properties defining a vertex ordering can affect the complexity of recognizing graphs in the class and we look at situations in which vertex orderings can help us in designing efficient algorithms for optimization problems. We give the first robust polynomial algorithm for the CLIQUE problem on IP-SEG graphs that is based on a vertex elimination ordering. We discuss a generalization of the approach to vertex and edge elimination orderings where induced neighborhoods belong to classes of graphs on which the CLIQUE problem is tractable. Further, we consider two different methods of combining vertex ordering properties and present examples of pairs of properties for which one of the methods leads to a more restricted class

of graphs. We show that there are cases where this leads to a class of graphs on which a generally hard optimization problem becomes tractable.

In Chapter V, we consider multiple graph classes and associated random generating models and we study how structural properties of the underlying graph can impact the effectiveness of mechanisms for facilitating or preventing decentralized consensus between agents positioned as vertices of a graph. The mechanisms being considered include communication, as well as introduction and varied placement of adversarial and trusted agents. We build on the work done in [56], where models of individual behavior were trained based on data from experiments with human subjects, by performing agent-based simulations over a wider range of graph-generating models, including geometric-based models of special graph classes like IP-SEG graphs. We consider varying levels of modifications to models of individual behavior and show that agents could be more successful in reaching decentralized consensus if they were to more closely follow the lead of neighbors who are known to be non-adversarial agents. In addition, we demonstrate that consensus rates can be increased, over all considered graph classes and random generating models, by ensuring that vertices of trusted agents induce a connected subgraph. When model modifications are combined with optimized placement of trusted agents, we observe compounded gains in consensus rates. Finally, we find that reaching decentralized consensus on classes like unit-disk and interval graphs, that are more likely to exhibit a community structure, is particularly challenging and we may need to consider alternative placements of trusted agents that prioritize coverage of inter-community edges.

CHAPTER III

Geometric Intersection Models: Generalizing Interval and Permutation Graphs

3.1 Introduction

A large number of important graph classes are defined or can be characterized by a geometric intersection model. These characterizations can arise from different geometric objects that are being intersected. Boxicity graphs (intersection graphs of d-dimensional rectangles), circular-arc graphs (intersection graphs of arcs along a circle), and string graphs (intersection graphs of curves in the plane) [50, 104] are just a few of many such examples. Two particularly well-studied examples are the classes of interval and permutation graphs [41]. In both of their respective models, the intersecting objects are line segments in the plane, with different restrictions imposed on their positions. In interval graphs, the line segments must lie on a single line, while in permutation graphs, their endpoints must lie on two separate parallel lines. Because of the similarity, it is natural to look for geometric intersection models that would generalize those of interval and permutation graphs. One approach is to have geometric objects that generalize line segments. In the model of *simple triangle* graphs (also known in the literature as *point-interval*) [23, 107], the intersecting objects are triangles, while in the model of *trapezoid* graphs (also referred to as *interval-interval*) [23, 26], the intersecting objects are trapezoids.

Another way of generalizing the models of interval and permutation graphs is to use the same kind of intersecting geometric objects - straight line segments - but reduce the restrictions on their possible positions. If we drop all restrictions, then we obtain the large class of SEG graphs - the intersection graphs of straight line segments in the plane [15]. However, many of the standard optimization problems, including COLORING [37], INDEPENDENT SET [76], and CLIQUE [17], remain NP-hard on SEG graphs, even if the representation as a set of segments is given as part of the input. Sub-classes of SEG graphs, with restrictions on the number of directions that line segments could have, have been studied, including grid intersection (or 2-DIR) and $k-$DIR graphs [59, 76]. Such models, however, do not generalize the model of permutation graphs, where segments can have any direction in the plane, except being parallel with the two lines.

We introduce a new model in which the intersecting objects remain straight line segments, but

they can either lie along one of two horizontal lines or go from one horizontal line to the other. This leads to two natural generalizations of both interval and permutation graphs, based on whether all horizontal segments lie on the same line [67, 68]. We formally define the models and graph classes in Section 3.2. In Section 3.3, we show that these classes have implicit representations. Unlike simple triangle and trapezoid graphs, the two new classes are not contained in the class of perfect graphs. However, in Section 3.4, we show that we are somewhat limited in how we can represent chordless cycles using the new model, which helps us identify some forbidden subgraphs for the graph classes. In Section 3.5 we demonstrate several neighborhood-related properties satisfied by some vertices of the new classes and identify additional forbidden subgraphs. In Section 3.6, we present polynomial algorithms for the CLIQUE, INDEPENDENT SET, and LONGEST CHORDLESS CYCLE problems, when the model is given as part of the input. We offer partial results with respect to the recognition problem in Section 3.7. In Section 3.8, we briefly discuss possible further generalizations of the new classes of graphs. Finally, in Section 3.9, we discuss some of the open questions on the new graph classes.

3.2 Preliminaries

Let $G = (V, E)$ be a graph with a vertex set $V = \{v_1, v_2, \ldots, v_n\}$. An *intersection model* of G is a family of sets $S_i,\ i = 1, 2, \ldots, n$, such that $(v_i, v_j) \in E$ if and only if $S_i \cap S_j \neq \emptyset$. We say that G is an *intersection graph* of the family of sets S_i.

Many classes of graphs can be defined or characterized as the intersection graphs of different kinds of families of sets. For example, the line graph $L(G)$ of a graph G is the intersection graph of edges of G (seen as two-element sets of edge endpoints), EPT graphs are defined as the intersection graphs of nontrivial simple paths (seen as sets of edges) in trees [51], and chordal graphs can be characterized as the intersection graphs of subtrees of trees [15]. More often, the intersecting sets correspond to geometric objects in a Euclidean space, including circular arcs, disks, triangles, d-dimensional rectangles and d-dimensional spheres. We refer to intersection models of this kind as *geometric intersection models*. In this chapter, we are primarily interested in geometric intersection models of straight line segments in the Euclidean plane. The two most prominent and extensively studied graph classes with such a model are interval and permutation graphs.

Definition 3.2.1. Interval graphs are the intersection graphs of intervals on the real line.

Definition 3.2.2. Permutation graphs are the intersection graphs of straight line segments having an endpoint on each of two parallel lines.

An example of a graph that is both interval and permutation, together with the respective geometric intersection models, is given in Figure 3.1. Note that we can obtain an equivalent definition of interval graphs by dropping the requirement that the line in question is *real* and by substituting *intervals* with *straight line segments*. This allows for multiple natural ways of simultaneously generalizing the geometric intersection models of interval and permutation graphs. These include *point-interval* (or *simple triangle*) and *interval-interval* (or *trapezoid*) graphs introduced by Corneil and Kamula [23], in which the geometric objects (or sets) corresponding to vertices are formed by combining multiple line segments.

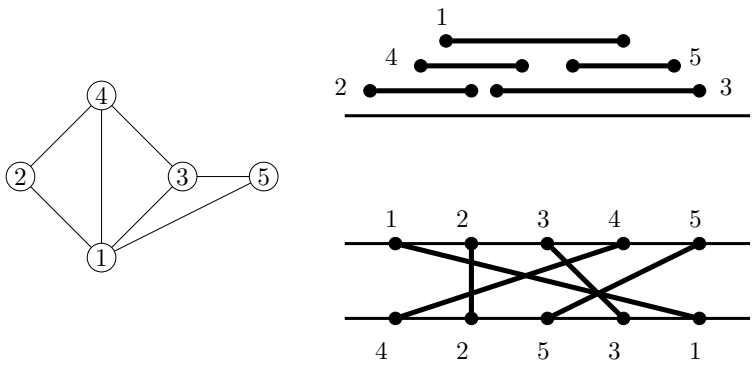


Figure 3.1: A graph that is both interval and permutation, together with a corresponding interval (top right) and permutation (bottom right) geometric intersection models.

In this work, we study generalization of the models of permutation and interval graphs in which the geometric objects corresponding to vertices remain individual line segments. We borrow the following piece of notation from [86]: for a given line segment s in the plane, by $t(s)$, $b(s)$, $l(s)$, and $r(s)$, we denote a top-, bottom-, left-, and rightmost point of s. We will reserve the use of this notation for situations where the extreme point of the segment is uniquely defined (as one of the segment's endpoints). We will also use the same notation when s represents a set of line segments.

Definition 3.2.3. Let $\mathscr{L}_1$ and $\mathscr{L}_2$ be two horizontal lines in the plane, with $\mathscr{L}_1$ lying above $\mathscr{L}_2$. We say that a straight line segment s is an *interval segment* if both its endpoints $l(s)$ and $r(s)$ lie on the same horizontal line $\mathscr{L}_i$. We say that s is a *permutation segment* if $t(s)$ lies on $\mathscr{L}_1$ and $b(s)$ lies on $\mathscr{L}_2$. Let $\mathscr{I}$ be a set of interval segments, $\mathscr{P}$ a set of permutation segments, and G the intersection

graph of the set of segments $\mathscr{I} \cup \mathscr{P}$. We call $M_G = \mathscr{I} \cup \mathscr{P}$ an *interval-permutation-segment* (*IP-SEG*) model of G. If all interval segments in $\mathscr{I}$ lie on the line $\mathscr{L}_1$, we call $M_G = \mathscr{I} \cup \mathscr{P}$ a *simple IP-SEG* model of G.

Definition 3.2.4. A graph G is an *IP-SEG graph* if it has an IP-SEG model. G is a *simple IP-SEG graph* if it has a simple IP-SEG model.

Note that under the above definitions, a simple IP-SEG model is also an IP-SEG model, which indicates that simple IP-SEG graphs are contained in IP-SEG graphs. We will later show that this containment is proper.

We know that other generalizations of permutation and interval graphs like simple triangle or trapezoid graphs remain perfect. One important characteristic that distinguishes IP-SEG and simple IP-SEG graphs is that they may contain odd holes, meaning they are not contained in the class of perfect graphs. Simple IP-SEG and IP-SEG models of a chordless cycle on five vertices are shown in Figure 3.2. It is easy to see how these models can be extended to models of larger chordless cycles.

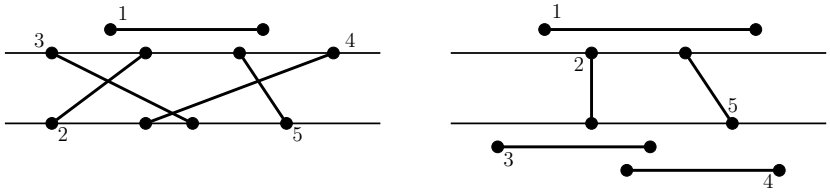


Figure 3.2: A simple IP-SEG model of a C_5 is shown on the left. An alternative IP-SEG model is given on the right. Note that the numbers shown next to line segments indicate the corresponding vertex labels from a natural labelling of C_5.

3.3 IP-SEG model representation

Each line segment s in the plane is uniquely determined by its two endpoints, each of which can be specified by their x and y coordinates. We will use the notation $s = (x_1(s), y_1(s), x_2(s), y_2(s))$. Thus, when we say we are given the model or representation of an IP-SEG graph G, we mean that for each vertex v of G we are given a four-tuple of numbers specifying the segment corresponding to v. When $x_i(s), y_i(s) \in \mathbb{R}$, the above is not a representation that characterizes IP-SEG graphs. In fact, it is a characterizing representation of the class of SEG graphs. However, we know that for IP-SEG graphs, the $y_i(s)$ coordinates identify the horizontal lines that the endpoints of a segment s

belong to, and without loss of generality, we may assume that $y_1(s), y_2(s) \in \{1,2\}$. Having $y_1(s) = y_2(s) = y(s)$ indicates that we are dealing with an interval segment and $y_1(s) \neq y_2(s)$ means we have a permutation segment. In addition, we may assume that for each permutation segment p the first endpoint is always positioned on $\mathscr{L}_\infty$ i.e. $y_1(p) = 1$ and $y_2(p) = 2$. Further, we may assume that for each interval segment, its left endpoint comes first, i.e. for $i = (x_1(i), y(i), x_2(i), y(i))$, we have $x_1(i) \leq x_2(i)$. We can also show that the range of values that $x_1(i)$ and $x_2(i)$ can take, can be limited to a finite set of integers.

Recall that in the model of interval graphs, whether two intervals share a non-empty intersection depends entirely on the relative ordering of their endpoints along the horizontal line. Similarly, whether two permutation segments of a permutation graph representation intersect depends entirely on the relative orderings of their endpoints along $\mathscr{L}_1$ and $\mathscr{L}_2$. A similar observation can be made for IP-SEG graphs. In particular, for two segments s and q, the following holds:

i) If s and q are two interval segments, they intersect if and only if

   - $y(s) = y(q)$ (s and q lie on the same horizontal line), **and**
   - $x_1(q) \leq x_1(s) \leq x_2(q)$ or $x_1(s) \leq x_1(q) \leq x_2(s)$ (s and q overlap).

ii) If s and q are two permutation segments, they intersect if and only if

   - $x_1(s) \leq x_1(q)$ and $x_2(q) \leq x_2(s)$, **or**
   - $x_1(q) \leq x_1(s)$ and $x_2(s) \leq x_2(q)$.

iii) If s is an interval segment and q is a permutation segment, they intersect if and only if s contains one of the endpoints of q, that is

   - $y(s) = 1$ and $x_1(s) \leq x_1(q) \leq x_2(s)$, **or**
   - $y(s) = 2$ and $x_1(s) \leq x_2(q) \leq x_2(s)$.

Thus, all we need to know to determine adjacency between two segments s and q from an IP-SEG model is to know what types of segments s and q are, on which line $\mathscr{L}_i$ each of their endpoints is located on, and what are the relative orderings of their endpoints on $\mathscr{L}_1$ and/or $\mathscr{L}_2$. The last part implies that the number of different values the x_i endpoint coordinates could take is bounded by the

maximum number of endpoints positioned on a single line and, by extension, the total number of endpoints. In other words, we may assume that in an IP-SEG model of a graph G on n vertices, for any line segment s in the model we have $x_1(s), x_2(s) \in \{1, 2, \ldots, 2n\}$.

With the above limitations on the range of x_i and y_i values, it follows that we need only $O(\log n)$ bits of information (the four-tuple specifying the corresponding segment) per vertex to properly represent any IP-SEG graph G, meaning that the number of IP-SEG graphs on n vertices is bounded by $2^{O(n \log n)}$. Furthermore, the $O(\log n)$ bits of information associated with vertices u and v are sufficient to test for adjacency between u and v in constant time. Therefore, similar to interval and permutation graphs, simple IP-SEG and IP-SEG graphs have an *implicit representation* [71, 104].

Like all classes with geometric intersection models, simple IP-SEG and IP-SEG are hereditary graph classes. Thus, they represent additional examples that conform with the implicit graph conjecture made by Kannan, Naor and Rudich in [71], which states that every hereditary class of graphs with $2^{O(n \log n)}$ graphs on n vertices has an implicit representation. On the other hand, the more general class of SEG graphs also satisfies the conditions of the conjecture, but has no known implicit representation [19]. The analogous representation using four-tuples does not work for SEG graphs, because real-valued coordinates can require an exponential number of bits.

Note that in the above discussion we have allowed for the possibility of segments sharing one or both endpoints. We know that permutation graphs are usually defined so that no two permutation segments share an endpoint. While this is not imposed as a requirement in the definition of interval graphs, the consecutive cliques arrangement [13] of an interval graph implies that every interval graph has a representation in which no two intervals share an endpoint. This is also the case for IP-SEG graphs.

Indeed, suppose that we have an IP-SEG model of a graph G in which line segments share endpoints and let e be one of those endpoints. Without loss of generality, we may assume e lies on $\mathscr{L}_1$. Let P be the set of permutation segments having e as an endpoint. Similarly, let I_L and I_R be the sets of interval segments having e as a left and right endpoint, respectively. We can modify the IP-SEG model of G so that e is no longer a shared endpoint, in the following way. Use the open interval i_e around e that does not contain any other endpoints. In it, arrange the $\mathscr{L}_1$ endpoints of permutation segments in P in a reverse order from the one their corresponding $\mathscr{L}_2$ endpoints have. In case some of the permutation segments also share an endpoint on $\mathscr{L}_2$, resolve the tie arbitrarily.

Then, we can extend the interval segments in I_L to the left, so that they all include the new endpoints of permutation segments in P, end at different endpoints, but their endpoints still remain within i_e. We can achieve the same thing with interval segments in I_R by extending them to the right. With this transformation, we ensure that permutation segments that shared e as an endpoint now either intersect between $\mathscr{L}_1$ and $\mathscr{L}_2$, or still have a shared endpoint on $\mathscr{L}_2$. In addition, all interval segments in I_L and I_R do contain the new endpoints of permutation segments P and all interval segments in I_L and I_R still contain the point e, meaning they have non-empty pairwise intersections. Finally, no new pairwise intersections are created as segments in $P \cup I_L \cup I_R$ are the only ones that have an endpoint in the open interval i_e.

Thus, the above transformation leads to a new IP-SEG model of G that has one fewer shared endpoint. By doing this repeatedly, we can obtain an IP-SEG model in which no two segments share an endpoint. The following lemma summarizes the results in this section.

Lemma 3.3.1. *Every IP-SEG graph has an IP-SEG model M_G such that:*

i) for every line segment $s \in M_G$, $x_1(s), x_2(s) \in \{1, 2, \ldots, 2n\}$ and $y_1(s), y_2(s) \in \{1, 2\}$, and

ii) no two segments in M_G share an endpoint.

From our discussion so far, it is clear that an IP-SEG graph may have more than one IP-SEG model. Nevertheless, when we deal with an IP-SEG graph G paired with a particular IP-SEG model M_G, there is a clear one-to-one correspondence between vertices in G and line segments in M_G. In such situations, for convenience, we will interchangeably use the notions of vertices and line segments, as well as the notions of induced subgraphs and sets of line segments.

3.4 Chordless cycles in IP-SEG graphs

In an IP-SEG model, two interval segments that do not lie on the same parallel line cannot intersect. Let G be a graph with an IP-SEG model consisting of interval segments only. If G is connected, then all of the interval segments in its IP-SEG model must lie on the same line. By extension, if G has more than one connected component, the interval segments of a single component C must lie on the same line. It is easy to see that the interval segments of a component C on $\mathscr{L}_2$ can be translated to $\mathscr{L}_1$, while ensuring that they do not intersect with other interval segments already on $\mathscr{L}_1$. This means that a graph G has an IP-SEG model consisting only of interval segments, if and

only if it has a simple IP-SEG model of only interval segments. In other words, G has an IP-SEG model consisting of only interval segments if and only if G is an interval graph. We can also make the analogous observation for permutation segments: a graph G has an IP-SEG model consisting of only permutation segments if and only if G is a permutation graph.

Interval graphs are chordal and therefore an IP-SEG model of a chordless cycle of length greater than three cannot consist exclusively of interval segments. Similarly, permutation graphs cannot contain an induced cycle on more than four vertices. Therefore, an IP-SEG model of a chordless cycle of length greater than four cannot consist exclusively of permutation segments. Figure 3.3 shows how a chordless cycle on four vertices may be represented using only permutation segments.

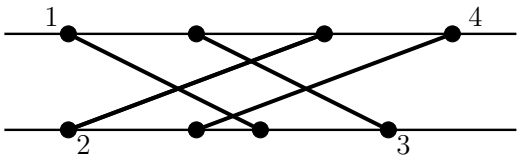


Figure 3.3: An IP-SEG model of a C_4 consisting of permutation segments only.

3.4.1 Interval and permutation arcs

We have already seen that a C_5 is a (simple) IP-SEG graph. It is easy to see how the models of C_5 in Figure 3.2 can be extended by adding more interval segments to represent larger chordless cycles. Our goal is to show that the ways in which such a cycle can be represented is, in a certain sense, limited and we will use that to identify examples of graphs that do not belong to the classes of simple IP-SEG and IP-SEG graphs.

Let u and v be vertices of an IP-SEG graph G corresponding to interval segments i_u and i_v, respectively, in a given IP-SEG representation of G. Further, suppose that i_v is fully contained in i_u. Then, u and v are adjacent and all neighbors of v must also be neighbors of u in G. In other words, $N[v] \subseteq N[u]$ must hold for the closed neighborhoods of v and u.

Consider a path $P_n = (v_1, v_2, \ldots, v_n)$. Clearly, $N[v_1] \subseteq N[v_2]$, $N[v_n] \subseteq N[v_{n-1}]$, and v_1 and v_n are the only vertices in P_n with the property of having a closed neighborhood that is contained within a closed neighborhood of another vertex. Now consider an IP-SEG representation of P_n using only interval segments. From the above discussion, it follows that an interval segment could be contained within another only if it corresponds to v_1 or v_n. We say that an IP-SEG representation of P_n using

only interval segments is an *active interval representation* if no interval segment is contained within another. Assuming that the interval segment corresponding to v_1 is the one with the leftmost left endpoint, it is easy to see that in an active interval representation of P_n, the left-to-right ordering of all left endpoints would coincide with the ordering of the vertices in P_n. The same is true for the left-to-right ordering of all right endpoints of interval segments. An example is shown in Figure 3.4.

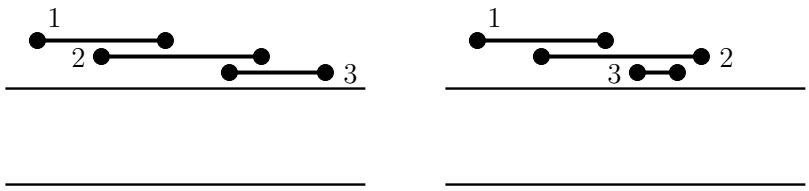


Figure 3.4: An active (left) and an inactive (right) interval representation of a P_3.

Suppose we are given an IP-SEG representation M_{C_n} of a chordless cycle $C_n = (v_1, v_2, \ldots, v_n)$, with $n \geq 4$. Let $s(v_i)$ denote the line segment in M_{C_n} corresponding to vertex v_i. Further, suppose that not all $s(v_i)$'s are permutation segments, i.e. that M_{C_n} consists of both interval and permutation segments. Let $S_{i,j} = (s(v_i), s(v_{i+1}), \ldots, s(v_j))$ be a maximal nonempty sequence of consecutive interval segments in IP(C_n), i.e. a sequence such that $s(v_{i-1})$ and $s(v_{j+1})$ are permutation segments and for each $k \in \{i, i+1, \ldots, j\}$, $s(v_k)$ is an interval segment (indices are modulo n). We call $S_{i,j}$ an *interval arc* of length $j - i + 1$. We can define the notion of a *permutation arc* of M_{C_n}, analogously. Using the above notation, any IP-SEG representation of a chordless cycle C_n, for $n \geq 5$, can be described as consisting of interval and permutation arcs. Note that because we define interval and permutation arcs as nonempty sequences of segments, the IP-SEG model of a C_4 shown in Figure 3.3 has 0 interval arcs and 0 permutation arcs. Clearly, all other IP-SEG models of chordless cycles which have segments of both types, must contain arcs of both types, and the number of interval arcs must equal that of permutation arcs. We also have the following result.

Lemma 3.4.1. *Let C_n be a chordless cycle, with $n \geq 4$. C_n has an IP-SEG representation with t interval arcs of lengths $\{l_1, l_2, \ldots, l_t\}$ if and only if $C_{n-n'}$, where $n' = \sum_{i=1}^{t}(l_i - 1)$, has an IP-SEG representation with t interval arcs, each of length* 1.

Proof. Let $S_{i,i+k}$ be an interval arc of a given IP-SEG representation of a chordless cycle C_n. Note that $S_{i,i+k}$, being formed by $k+1$ segments, must be an active interval representation of the path P_{k+1}. Suppose we take one interval segment s_j in $S_{i,i+k}$ and replace it with two partially overlapping

interval segments $s_{j'}$ and $s_{j''}$ such that $s_j = s_{j'} \cup s_{j''}$ and $s_{j'} \cap s_{j''}$ does not overlap any of the other segments in $S_{i,i+k}$. With this we are transforming an IP-SEG representation of C_n into an IP-SEG representation of C_{n+1}. Similarly, assuming $k \geq 1$, we can take two consecutive interval segments s_j and s_{j+1} in $S_{i,i+k}$ and replace them with one new segment s_{j*} such that $s_{j*} = s_j \cup s_{j+1}$. This allows us to transform an IP-SEG representation of C_n with an interval arc of length $k+1$ into an IP-SEG representation of C_{n-1} with an interval arc of length k. The more general result stated in the lemma follows from a repeated application of the above transformation. □

A natural question to ask is how many interval arcs an IP-SEG representation of a chordless cycle can have. Recall that in Figure 3.2 we saw one (simple) IP-SEG representation of a C_5 consisting of one interval arc and one permutation arc, as well as another IP-SEG representation consisting of two interval arcs and two permutation arcs. We also noted that a C_4 is the only example of a chordless cycle that can have no arcs. We will show that these are in fact the only three possibilities.

Lemma 3.4.2. *Let M_{C_n} be a simple IP-SEG model of a chordless cycle C_n, containing both interval and permutation segments. Then M_{C_n} must have exactly one interval arc.*

Proof. Suppose there exists a chordless cycle C_n with a simple IP-SEG model that has two interval arcs. Then, by Lemma 3.4.1, there exists a chordless cycle C_m, $m \leq n$, with a simple IP-SEG model that has two one-segment interval arcs I_1 and I_2. Without loss of generality, we may assume that I_1 lies to the left of I_2 on $\mathscr{L}_1$. Let $P_1 = (p_{1,1}, p_{1,2}, \ldots, p_{1,q})$ and $P_2 = (p_{2,1}, p_{2,2}, \ldots, p_{2,r})$ be the two permutation arcs so that $p_{1,1}$ and $p_{2,1}$ are the permutation segments that intersect I_1 and the intersection point $p_{2,1} \cap I_1$ is to the right of $p_{1,1} \cap I_1$. Consider the points $z_0 = p_{1,1} \cap I_1$, $z_q = p_{1,q} \cap I_2$, and $z_k = p_{1,k} \cap p_{1,k+1}$, $1 \leq k < q$. The straight-line segments connecting z_i with z_{i+1}, $0 \leq i \leq q$, form a broken line $B(P_1)$ connecting z_0 with z_q. Note that $B(P_1)$ is a continuous curve connecting z_0 and z_q that is bounded between $\mathscr{L}_1$ and $\mathscr{L}_2$. As such, one of its constituting straight-line segments must intersect $p_{2,1}$ (see Figure 3.5). Since each of these line segments must be contained within some permutation segment of P_1, there must be a permutation segment $p_{1,x}$ that intersects $p_{2,1}$. However, $p_{1,x}$ and $p_{2,1}$ correspond to two non-consecutive vertices of a chordless cycle that cannot be adjacent, which is a contradiction. Therefore, a C_n cannot have a simple IP-SEG model with two interval arcs. Furthermore, if a C_n could have a simple IP-SEG model with $k \geq 3$ interval arcs, we could turn this into a simple IP-SEG model with $k-1$ interval arcs by collapsing one of the

interval arcs into a single point and effectively merging two permutation arcs into one. Therefore, a C_n cannot have a simple IP-SEG model with more than one interval arc. □

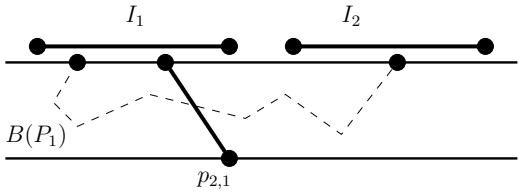


Figure 3.5: A simple IP-SEG representation of a chordless cycle cannot consist of two interval arcs because the broken line $B(P_1)$ induced by the first permutation arc P_1 will have to intersect a segment of the other permutation arc P_2.

Lemma 3.4.3. *Let M_{C_n} be an IP-SEG model of a chordless cycle C_n, containing both interval and permutation segments. Then M_{C_n} must have either exactly one interval arc, or exactly two interval arcs positioned on separate horizontal lines.*

Proof. Given Lemma 3.4.2, we only need to consider "proper" IP-SEG models having at least one interval arc on each horizontal line. Suppose there exists a chordless cycle C_n with an IP-SEG model that has two interval arcs I_1 and I_2 on $\mathscr{L}_1$ and one interval arc I_3 on $\mathscr{L}_2$. As in the proof of Lemma 3.4.2, we may assume that each of these are one-segment interval arcs. Let $P_1 = (p_{1,1}, p_{1,2}, \ldots, p_{1,q})$ be the permutation arc between I_1 and I_2, $P_2 = (p_{2,1}, p_{2,2}, \ldots, p_{2,r})$ the permutation arc between I_2 and I_3 and $P_3 = (p_{3,1}, p_{3,2}, \ldots, p_{3,s})$ the permutation arc between I_1 and I_3. Further, assume that $p_{1,1}$ and $p_{3,1}$ are the permutation segments of P_1 and P_3, respectively, intersecting with I_1.

If the intersection point $p_{1,1} \cap I_1$ is to the left of $p_{3,1} \cap I_1$ (left half of Figure 3.6), then the broken line $B(P_1)$ induced by the intersection points of permutation segments of P_1 (see proof of Lemma 3.4.2) must intersect $p_{3,1}$. This would imply an intersection between two permutation segments that correspond to non-consecutive vertices in C_n, a contradiction.

Suppose the intersection point $p_{3,1} \cap I_1$ is to the left of $p_{1,1} \cap I_1$ (right half of Figure 3.6) and consider the endpoint of $p_{1,1}$ that lies on $\mathscr{L}_2$. It cannot lie within I_3 as $p_{1,1}$ and I_3 do not correspond to consecutive vertices in the chordless cycle. However, if it were to lie to the left or right of I_3, then $B(P_3)$ or $B(P_2)$ would need to intersect with $p_{1,1}$, both of which lead to a contradiction. Therefore, a C_n cannot have an IP-SEG model with two interval arcs on one horizontal line and one interval

arc on the other. Recalling the observation about collapsing interval arcs on the same line, this also implies that an IP-SEG model of a chordless cycle C_n cannot have more than one interval arc per horizontal line. □

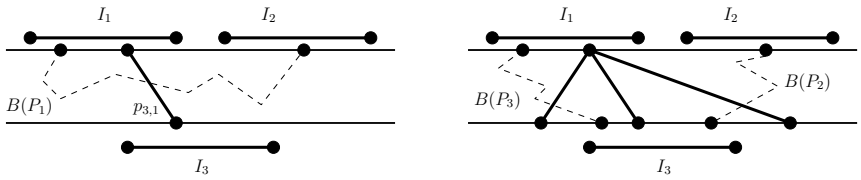


Figure 3.6: An IP-SEG representation of a chordless cycle cannot consist of two interval arcs on $\mathscr{L}_1$ and one on $\mathscr{L}_2$ because: (1) the broken line $B(P_1)$ induced by the permutation arc P_1 will have to intersect a segment of the permutation arc P_3 (model on the left) or (2) a segment of the permutation arc P_1 would have to intersect either $B(P_2)$, $B(P_3)$, or I_3 (model on the right).

3.4.2 Permutation arcs are always short

From our discussion of the IP-SEG models of C_5 shown in Figure 3.2 and from the proof of Lemma 3.4.1, we know that interval arcs can have arbitrarily large lengths, assuming the corresponding chordless cycle is large enough. This, however, is not the case for permutation arcs. First, consider a simple IP-SEG model of a chordless cycle C_n with an interval arc I and a permutation arc $P = (p_1, \ldots, p_k)$, such that p_1 intersects I to the left of p_k. It is easy to see that $k \geq 2$ is a tight lower bound. Suppose $k \geq 5$ and let O_1 and O_2 be the two permutations of $(1, \ldots, k)$ that define the relative ordering of endpoints of permutation segments in P along the horizontal lines $\mathscr{L}_1$ and $\mathscr{L}_1$, respectively. Then p_1 and p_k will correspond to nonconsecutive vertices of C_n and 1 will have to appear before k in both O_1 and O_2. Since p_1 and p_k are the only permutation segments intersecting I, 1 and k must appear consecutively in O_1. Further, p_3 cannot intersect neither p_1 nor p_k. Also, p_3 cannot have both of its endpoints to the left of p_1 because one of the segments $p_4, p_5, \ldots, p_{k-1}$ (and the broken line they form) would have to cross p_1. Similarly, p_3 cannot be be entirely to the right of p_k. The only other potential option is to place p_3 between p_1 and p_k, but to achieve this we would need for both O_1 and O_2 to contain the sub-ordering $(1,3,k)$, which contradicts 1 and k being consecutive in O_1. Therefore, the length k of a permutation arc in a simple IP-SEG model of C_n is bounded by $2 \leq k \leq 4$.

Now, consider an IP-SEG model of C_n having two interval arcs I_1 and I_2, with I_1 being the one along $\mathscr{L}_1$. Let the two permutation arcs be $P = (p_1, \ldots, p_k)$ and $Q = (q_1, \ldots, q_l)$, with $t(p_1), t(q_1) \in$

I_1, $b(p_k), b(q_l) \in I_2$, and $t(p_1)$ being to the left of $t(q_1)$. Since segments of P cannot intersect with segments of Q, $b(p_1)$ cannot be to the right of $b(q_l)$. If $b(p_1)$ lies on I_2 then we must have $k = 1$. Suppose that $k \geq 2$. Then $b(p_1) < l(I_2) < b(p_k)$ and, by symmetry, $t(p_k) < l(I_1) < t(p_1)$. This implies that p_1 intersects p_k, which is only possible if they correspond to consecutive vertices in C_n, i.e. if $k = 2$. Therefore, in an IP-SEG model of C_n that has two arcs of each kind, a single permutation arc can consist of either one or two segments. Figure 3.7 summarizes all possible configurations of chordless cycles that contain both interval and permutation segments.

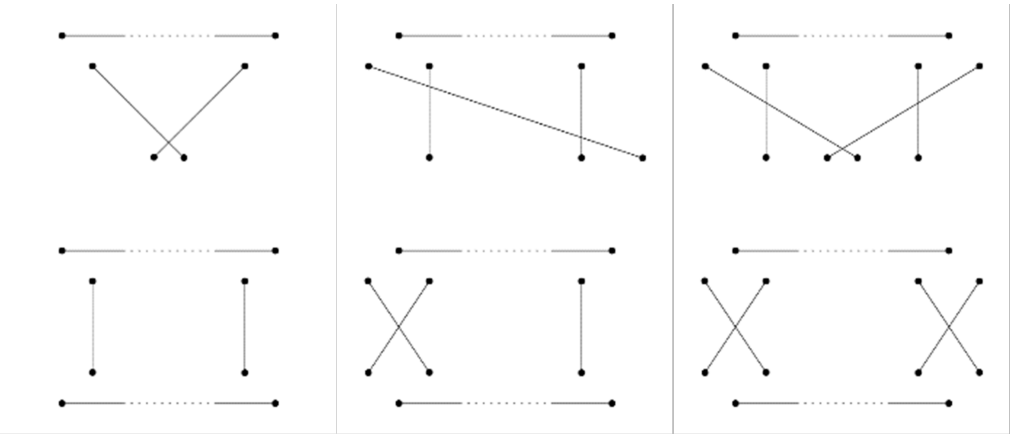


Figure 3.7: The possible IP-SEG configurations of C_n with one arc of each kind (top) and with two arcs of each kind (bottom). Dashed lines indicate that interval arcs can have arbitrary length. Note that the left-to-right symmetric equivalents of the top middle and bottom middle configurations are also possible.

Knowing the limited ways in which we can represent a chordless cycle in an IP-SEG model, we can describe the first family of forbidden induced subgraphs for the classes of simple IP-SEG and IP-SEG graphs. Consider the graph $G_{7,2}$ on 21 vertices formed by a cycle $C_7 = (v_1, v_2, \ldots, v_7)$ and vertices w_i and z_i, $1 \leq i \leq 7$, such that w_i is pendant to v_i and z_i is pendant to w_i. A simple IP-SEG model M_{C_7} of the C_7 in $G_{7,2}$ must consist of one interval arc I and one permutation arc P. As P cannot have length greater than 4, I must be composed of at least three segments. Then there exists a vertex v_t corresponding to an *interior* interval segment s_t of I, i.e. a segment such that $l(I) < l(s_t)$ and $r(s_t) < r(I)$. Moreover, s_t must fully lie between the endpoints of the broken line $B(P)$ induced by the permutation arc. Consequently, the segment corresponding to v_t's pendant neighbor w_t cannot be a permutation segment as it would intersect with at least one segment in P. This, combined with the fact that s_t's ends are overlapped by the two neighboring interval segments,

implies that w_t must correspond to an interval segment that is fully contained in s_t. However, w_t has a neighbor z_t that it does not share with v_t, a contradiction. Therefore, $G_{7,2}$ is not a simple IP-SEG graph. Note that this would be true for any graph $G_{n,2}$ constructed in an analogous way from a cycle C_n with $n \geq 7$. One can easily verify that $G_{7,2}$ is also a minimal non-member of simple IP-SEG graphs, under vertex removal. Note that the above argument showing that $G_{7,2}$ is not a simple IP-SEG graph does not require the vertices z_i to be pairwise nonadjacent, meaning that any combination of edges (z_i, z_j) leads to other examples of graphs that are not simple IP-SEG.

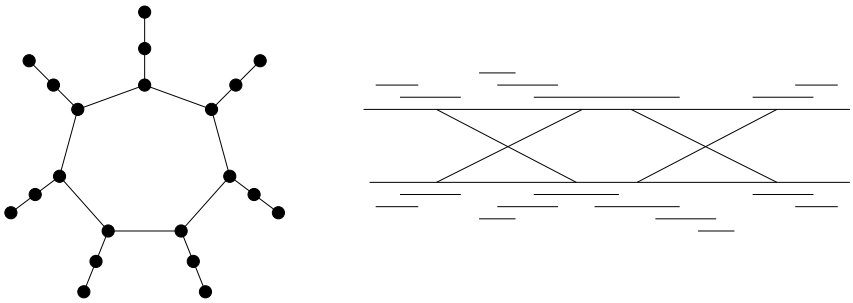


Figure 3.8: The graph $G_{7,2}$ (left) is not a simple IP-SEG graph, but is an IP-SEG graph. One possible IP-SEG model of $G_{7,2}$ is shown on the right.

On the other hand, $G_{7,2}$ and $G_{8,2}$, can be represented using an IP-SEG model (for $G_{7,2}$ see Figure 3.8), which demonstrates that the class of simple IP-SEG graphs is properly contained in IP-SEG. However, this is not true for graphs $G_{n,2}$ with $n \geq 9$. We know that in an IP-SEG model M_{C_n} of C_n which has two interval and two permutation arcs, the permutation arcs cannot consist of more than two segments. Thus, when $n \geq 9$, at least one interval arc of M_{C_n} would need to be of length at least three and at least one vertex v_t would have to correspond to an interior interval segment. This would again lead to a contradiction when trying to assign segments to w_t and z_t.

3.5 Neighborhood properties and other forbidden induced subgraphs of IP-SEG graphs

The observed limitations on IP-SEG models of chordless cycles that use segments of both types allow us to make some inferences on the kinds of neighborhoods a vertex of an IP-SEG graph may have. Let G be an IP-SEG graph and M_G be one of its IP-SEG models. Suppose v is a vertex of G such that $N(v)$ contains a $C_n = \{c_1, c_2, \ldots, c_n\}$ as an induced subgraph, with $n \geq 5$. A short inspection of the possible C_n configurations shown in Figure 3.7 suffices to see that v cannot correspond to a permutation segment in M_G. Without loss of generality, we may assume that the

interval segment $i(v)$ that v corresponds to lies on $\mathscr{L}_1$. Since $n \geq 5$, we know that there has to be at least one vertex c_k that corresponds to an interval segment $i(c_k)$. Because an interval segment cannot intersect interval segments lying on a different horizontal line, $i(c_k)$, as well as any other interval segments corresponding to vertices of C_n in M_G, must lie on $\mathscr{L}_1$. If u is a vertex different from v, such that $N(u)$ contains the same $C_n = \{c_1, c_2, \ldots, c_n\}$, then u must also correspond to an interval segment $i(u)$ lying on $\mathscr{L}_1$ in M_G. Because C_n must contain at least one vertex c_l corresponding to a permutation segment in M_G, $i(v)$ and $i(u)$ must have a non-empty intersection (at the very least, containing the top endpoint of c_l's permutation segment). Let $W_{n,2}$ be the graph formed by adding a single vertex v to a wheel graph W_n and making v a false twin of the apex of W_n. We call $W_{n,2}$ a *2-apex wheel graph*. The above discussion implies that 2-apex wheel graphs $W_{n,2}$, with $n \geq 5$, are another set of graphs that do not belong to the class of IP-SEG graphs. The 2-apex wheel graph $W_{5,2}$ shown in Figure 3.9) is a minimal non-member of IP-SEG graphs.

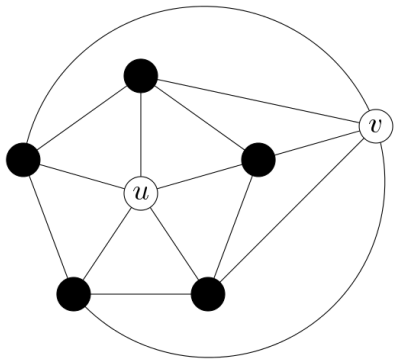


Figure 3.9: The 2-apex wheel graph $W_{5,2}$ formed by taking the wheel graph on six vertices W_5 and adding to it a false twin v of the wheel apex u.

We can say something even stronger about the interval segment $i(v)$ in the above discussion. If C_n, in M_G, is represented using the first configuration (see Figure 3.7), its interval arc I will need to be of length at least 3 and contain at least one interior interval segment, which $i(v)$ will need to fully contain. Similarly, if we are dealing with the second configuration, the length of the interval arc I will need to be at least two and, to ensure that $i(v)$ intersects all of the permutation segments, $i(v)$ will have to fully contain the leftmost interval segment of I. Finally, in the third configuration, which will have a nonempty interval arc I, in order to intersect all permutation segments, $i(v)$ will need to fully contain all the segments of I. In short, if a vertex v of an IP-SEG graph G contains a chordless cycle in its neighborhood, then in any IP-SEG model of G, v necessarily corresponds

to an interval segment $i(v)$ that contains at least one other interval segment. Having in mind that C_n, for $n \geq 5$, is not a permutation graph, we have the following, significantly more general, result regarding the neighborhoods of vertices of IP-SEG graphs.

Lemma 3.5.1. *Let G be an IP-SEG graph and let M_G be one of its IP-SEG models. If v is a vertex of G with a corresponding line segment $s(v)$ in M_G such that:*

a) $s(v)$ is an interval segment not containing any other interval segments, or

b) $s(v)$ is a permutation segment such that no interval segment intersecting $s(v)$ is contained within another interval segment,

then $N[v]$ must induce a permutation graph.

Proof. We will prove this result by showing how the line segments corresponding to vertices in $N[v]$ can be mapped from M_G to another IP-SEG model consisting exclusively of permutation segments, while maintaining the same set of segment pairs having non-empty intersections.

a) Suppose $s(v)$ is an interval segment, lying on $\mathscr{L}_1$, and such that it does not contain any other interval segments. We can split the set of segments of M_G corresponding to vertices of $N[v]$ into the following 4 groups:

- $\mathscr{P}$: all permutation segments,
- $\mathscr{I}_L$: interval segments i that overlap $s(v)$ on the left, i.e. such that $l(i) < l(s(v))$ and $r(i) \in s(v)$,
- $\mathscr{I}_R$: interval segments i that overlap $s(v)$ on the right, i.e. such that $l(i) \in s(v)$ and $r(i) > r(s(v))$,
- $\mathscr{I}_C$: interval segments i that contain $s(v)$ (we include $s(v)$ itself in this set).

Let L^* be a point on the lower horizontal line $\mathscr{L}_2$ satisfying $L^* < b(p)$, for all $p \in \mathscr{P}$. Let R^* be a right-hand side analog of L^*. Consider the following mapping τ_a:

$$\tau_a(q) = \begin{cases} q, & q \in \mathscr{P} \\ \text{permutation segment } p \text{ with } t(p) = r(q) \text{ and } b(p) = L^*, & q \in \mathscr{I}_L \\ \text{permutation segment } p \text{ with } t(p) = l(q) \text{ and } b(p) = R^*, & q \in \mathscr{I}_R \cup \mathscr{I}_C \end{cases}$$

Figure 3.10 illustrates why τ_a produces a valid permutation representation of $G[N[v]]$ that maintains the original set of segment pairs with non-empty intersections from M_G.

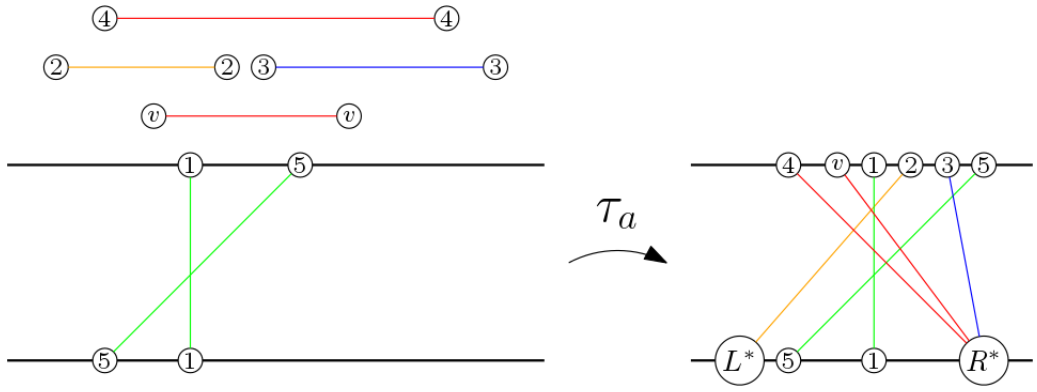


Figure 3.10: An IP-SEG representation of the neighborhood of vertex v, with v represented as an interval segment not contained by another (model on the left) and how that representation can be mapped to another (model on the right) consisting entirely of permutation segments.

b) Now suppose that $s(v)$ is a permutation segment such that no interval segment intersecting $s(v)$ is contained within another interval segment. In part *a)*, our task was simpler in that we were able to come up with an appropriate transformation that kept all permutation segments in their original positions. However, such a transformation is not always possible when $s(v)$ is a permutation segment. Consider a vertex u in $N[v]$ corresponding to an interval segment $i(u) \in \mathscr{L}_1$ such that $l_1^* < l(i(u))$ and $r(i(u)) < r_1^*$, where $l_1^* = min\{t(p) : p \in \mathscr{P}\}$ and $r_1^* = max\{t(p) : p \in \mathscr{P}\}$. Further, suppose that $i(u)$ intersects with at least one permutation segment p_l such that $l_1^* < t(p_l) < t(s(v))$ and at least one permutation segment p_r such that $r_1^* > t(p_r) > t(s(v))$. It is easy to see that the only way to map $i(u)$ to a permutation segment $p(u)$ that intersects both p_l and p_r is if $p(u)$ intersects a permutation segment ending at l_1^* or r_1^*, which would introduce at least one new pair of segments with a non-empty intersection.

There is, however, a way to rearrange the endpoints of permutation segments, so that all interval segments corresponding to vertices of $N[v]$ in M_G can be mapped to new permutation segments, while maintaining the same set of segment pairs with non-empty intersections. We divide the endpoints of segments, both interval and permutation, intersecting $s(v)$ into 4 groups:

- L_T: endpoints e on $\mathscr{L}_1$ such that $e < t(s(v))$,
- R_T: endpoints e on $\mathscr{L}_1$ such that $e > t(s(v))$,

- L_B: endpoints e on $\mathscr{L}_2$ such that $e < b(s(v))$,
- R_B: endpoints e on $\mathscr{L}_2$ such that $e > b(s(v))$.

Note that for each permutation segment p intersecting $s(v)$ we either have: 1) $t(p) \in L_T$ and $b(p) \in R_B$, or 2) $t(p) \in R_T$ and $b(p) \in L_B$. Similarly, for each interval segment i intersecting $s(v)$ we either have: 1) $l(i) \in L_T$ and $r(i) \in R_T$, or 2) $l(i) \in L_B$ and $r(i) \in R_B$.

The first step of our transformation is to reverse the ordering of endpoints in L_T. More formally, for $e \in L_T$, let $l(e)$ denote the line segment that has e as one of its endpoints. Further, let $e_1 < e_2 < \cdots < e_{|L_T|}$ be the ordering of endpoints in L_T. Then, for each $k \in \{1,2,\ldots,|L_T|\}$, we make e_k the new endpoint of segment $l(e_{|L_T|-k+1})$ in L_T. In the second step of the transformation, we perform an analogous reversal of the endpoints in R_B. The third step amounts to swapping the sets R_T and L_B. Namely, let $f_1 < f_2 < \cdots < f_{|R_T|}$ and $g_1 < g_2 < \cdots < g_{|L_B|}$ be the endpoints in R_T and L_B respectively. We translate the endpoints in R_T from $\mathscr{L}_1$ to $\mathscr{L}_2$, so that their relative left-to-right ordering is preserved and $f_{|R_T|}$ is positioned to the left of $b(s(v))$. Similarly, we translate the endpoints in L_B from $\mathscr{L}_2$ to $\mathscr{L}_1$, so that their relative left-to-right ordering is preserved and g_1 is positioned to the right of $t(s(v))$.

Steps one and two do not change the type of a line segment - permutation (interval) segments remain permutation (interval) segments. The swap of endpoints in step three, on the other hand, moves one endpoint of each interval segment to a different horizontal line, in effect mapping it to a permutation segment. It is easy to confirm that once the three steps are completed, pairs of permutation that intersected before, intersect now too. Interval segments originally lying on $\mathscr{L}_1$, now become permutation segments entirely to the left of $s(v)$ which, because of the re-positioning of endpoints of the original permutation segments, now have the desired intersections. The same is true for the interval segments originally lying on $\mathscr{L}_2$, which now lie to the right of $s(v)$. The only issue is that the permutation segments that interval segments are mapped to, now do not intersect $s(v)$. However, we can easily resolve this by re-positioning $s(v)$ so that it intersects all newly formed permutation segments. We illustrate the full transformation in Figure 3.11.

□

Any non-empty IP-SEG model must include an interval segment not containing any other interval segments. One such example is the interval segment on $\mathscr{L}_1$ that has the leftmost right endpoint.

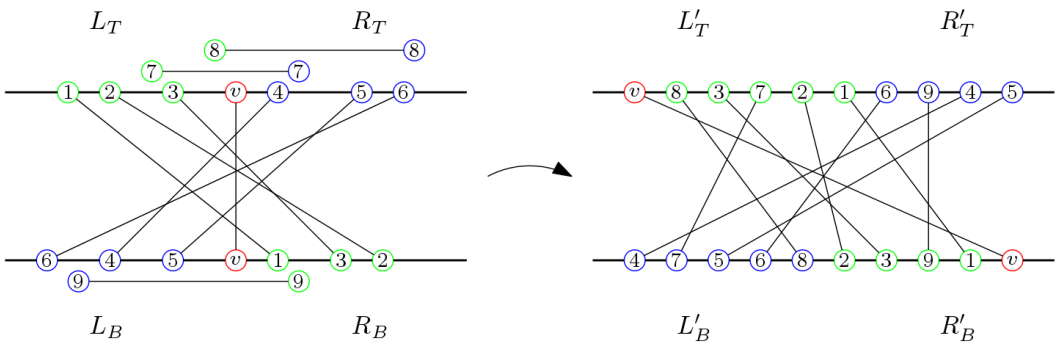


Figure 3.11: An IP-SEG representation of the neighborhood of vertex v - with v represented as a permutation segment (left) - and how that representation can be mapped to another (right) consisting entirely of permutation segments. The mapping reverses the orderings of endpoints in L_T and R_B. Then, it swaps the elements of sets R_T and L_B, while preserving the respective orderings. Finally, it re-positions the segment corresponding to vertex v so that it intersects all other segments.

Thus, we have the following simple corollary of Lemma 3.5.1.

Corollary 3.5.1. *Any IP-SEG graph G must have at least one vertex v such that $G[N[v]]$ is a permutation graph.*

Note that Corollary 3.5.1 does not constitute a characterization of IP-SEG graphs. All vertices of the graph $G_{n,2}$ (described in the previous section) have closed neighborhoods that induce permutation graphs, but $G_{n,2}$ is not an IP-SEG graph when $n \geq 9$. Nevertheless, Corollary 3.5.1 gives us another family of graphs that are not IP-SEG, namely graphs that do not have vertices whose neighborhood induces a permutation graph. There is one example set of such graphs that is particularly easy to construct. Let $k \geq 5$ and let $C_{k,2}$ be the graph on $n = 2k$ vertices formed by taking the union of two copies of C_k and adding all possible edges that go between vertices of different copies. Clearly, for each vertex $v \in C_{k,2}$, $G[N[v]]$ contains an induced chordless cycle of length greater than 4 and therefore cannot be a permutation graph. Note that $C_{k,2}$ is not a minimal non-member of IP-SEG graphs under vertex removal, as it contains instances of the 2-apex wheel graph $W_{k,2}$ as induced subgraphs. We will see another, more significant consequence of Corollary 3.5.1, in the next chapter when we discuss a robust algorithm for the *CLIQUE* problem based on a vertex elimination ordering.

We conclude this section with an observation regarding asteroidal triples. The neighborhood of a vertex v in an IP-SEG graph G can contain an asteroidal triple (a_1, a_2, a_3), but in any valid IP-SEG model of G, v will have to correspond to an interval segment containing another as permutation

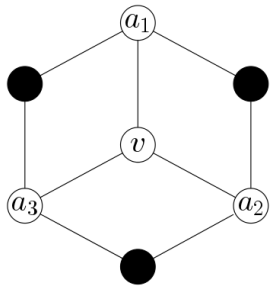


Figure 3.12: An example of a graph that is not IP-SEG, in which a vertex v is adjacent only to the vertices of an asteroidal triple (a_1, a_2, a_3).

graphs cannot contain asteroidal triples. However, we cannot have a vertex u whose only neighbors are a_1, a_2, and a_3. An example of such a graph, that is not IP-SEG, is shown in Figure 3.12. The following lemma summarizes the result.

Lemma 3.5.2. *Let G be an IP-SEG graph containing an asteroidal triple (a_1, a_2, a_3) and a vertex u such that $a_1, a_2, a_3 \in N(u)$. Then, there exists $i \in \{1,2,3\}$ such that $N(a_i) \subseteq N(u)$.*

Proof. Let $M(G)$ be an IP-SEG model of G and p be a permutation segment in $M(G)$. We say that another segment $s \in M(G)$ (interval or permutation) *lies to the left (right) of* p, if any endpoints of s on $\mathscr{L}_1$ lie to the left (right) of $t(p)$ and similarly, any endpoints of s on $\mathscr{L}_2$ lie to the left (right) of $b(p)$. Further, given two segments $s_1, s_2 \in M(G)$, we say that p *separates* s_1 and s_2 if one of the two segments lies to the left, and the other to the right, of p. For a vertex x of G we denote by $s(x)$ the line segment corresponding to x in $M(G)$.

We consider two cases. First, suppose that at least one of $s(a_1), s(a_2)$, and $s(a_3)$ is a permutation segment. Without loss of generality, we may assume that this includes $s(a_1)$. It is easy to see that $s(a_1)$ cannot separate $s(a_2)$ and $s(a_3)$, as any path between $s(a_2)$ and $s(a_3)$ will need to contain a vertex whose corresponding segment intersects $s(a_1)$. Thus, $s(a_1), s(a_2)$, and $s(a_3)$, up to horizontal and vertical symmetry and swapping $s(a_2)$ with $s(a_3)$, can have only one of the three potential arrangements shown in the top half of Figure 3.13. Note that in arrangement 1), there is no way to place $s(u)$ so that it intersects all three segments. Therefore, we actually cannot have this arrangement in $M(G)$. The arrangements 2) and 3) are possible in $M(G)$, but the only way to place $s(v)$ so that it intersects all three segments is as an interval segment on the top line containing the $s(a_2)$ segment. On the other hand, if each of $s(a_1), s(a_2)$, and $s(a_3)$ is an interval segment, we essentially

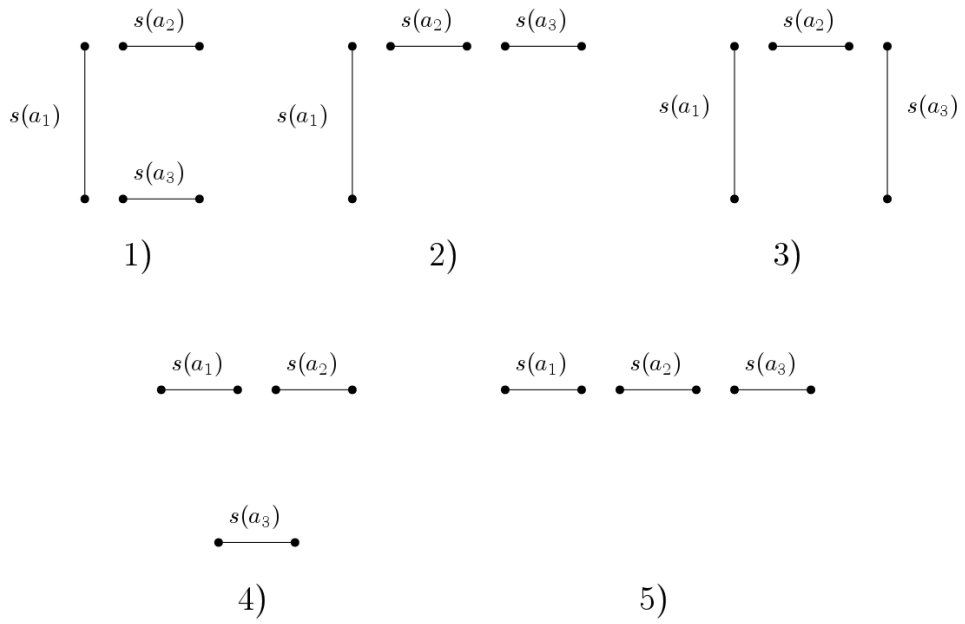


Figure 3.13: Possible configurations of segments corresponding to three independent vertices, up to symmetry and segment relabeling.

have two arrangements to consider, shown in the bottom half of Figure 3.13. In arrangement 4), we cannot place $s(u)$ so that it intersects all three of $s(a_1), s(a_2)$, and $s(a_3)$, so this arrangement cannot occur in $M(G)$. Arrangement 5) remains possible, but as before, the only way to place $s(u)$ is as an interval segment on $\mathscr{L}_1$ containing $s(a_2)$. Since $M(G)$ is an IP-SEG model of G and $s(u)$ and $s(a_2)$ must be interval segments with $s(u)$ containing $s(a_2)$ in $M(G)$, we must have that $N(a_2) \subseteq N(u)$.

□

3.6 Polynomial algorithms for optimization problems when the IP-SEG model is given

The respective geometric intersection models of interval and permutation graphs, imply very natural polynomial algorithms for solving several important optimization problems, including clique, independent set, and graph coloring. For example, given the model of an interval graph, we can find a maximum independent set by using a greedy algorithm which at each step selects the interval with the leftmost right endpoint, while removing all other intervals that intersect it from consideration. Given the model of a permutation graph, we can easily find a maximum clique by recovering the defining permutation of the graph from the model and finding the longest decreasing subsequence in it. These algorithms are not of great practical importance, as there exist linear-time algorithms for

the respective problems on larger graph classes, such as chordal and comparability graphs, which do not require a model as part of the input. Nevertheless, such model-based algorithms point us to an initial direction in the study of optimization problems on the new classes of simple IP-SEG and IP-SEG graphs. In particular, we look at the following three optimization problems when the IP-SEG model is given as part of the input: CLIQUE, INDEPENDENT SET, and LONGEST CHORDLESS CYCLE.

3.6.1 The CLIQUE problem

Suppose that C is a clique in an IP-SEG graph G with a given IP-SEG model M_G. Since interval segments lying on different parallel lines cannot intersect, all interval segments in C, if any, must lie on a single line. Without loss of generality, we may assume that line is $\mathscr{L}_1$. Denote by C_{int} the set of interval segments in C and by C_{prm} the set of permutation segments in C.

Consider the case when $C_{\text{int}} \neq \emptyset$ and let s_C be the intersection of interval segments in C_{int}. As such, s_C is fully contained in each interval segment in C_{int} and it must contain the top endpoint of each permutation segment in C_{prm}. In addition, the endpoints of s_C must be endpoints of one or more interval segments in C_{int}. Figure 3.14 provides an illustration.

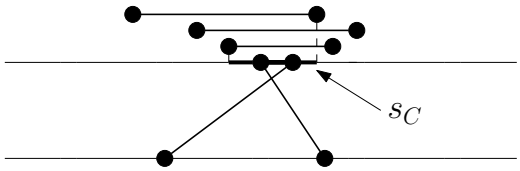


Figure 3.14: An IP-SEG representation of a clique on 5 vertices.

If we knew what s_C was, it would not be difficult to recover a clique of maximum size. First, set C_{int} to be the set of interval segments fully containing s_C. Then, identify all permutation segments that have an endpoint in s_C and find a clique of maximum size C_{prm} on the permutation graph they induce. Note that, since there may be multiple possible options for C_{prm}, there may be multiple cliques of maximum size that correspond to s_C. Nevertheless, the above procedure will recover a maximum clique, when we know s_C.

This leads to a simple polynomial algorithm for the clique problem, when the IP-SEG model is known [67]. First, identify the maximum clique on the graph G_p induced by the permutation segments of G. Then, go over all pairs of endpoints (s_1, s_2) of interval segments (s_1 and s_2 may

belong to different segments) along $\mathscr{L}_1$ and identify the maximum clique formed by interval segments containing the interval $s_C = [s_1, s_2]$ and permutation segments with an endpoint in s_C. Repeat the same for pairs of endpoints along $\mathscr{L}_2$. The clique of largest size found in this procedure is a maximum clique of G.

Algorithm 1 CLIQUE on IP-SEG graphs (given the model)

Clique(G, M_G)

```
1: C_p: maximum clique of the graph induced by all permutation segments of G
2: C_max = C_p
3: for i ∈ {1,2} do
4:    for each pair of endpoints (s_1, s_2) on L_i do
5:       C_int: interval segments on 𝒮_i containing the interval [s_1, s_2]
6:       C_prm: maximum clique formed by permutation segments with an endpoint in [s_1, s_2]
7:       C = C_int ∪ C_prm
8:       if |C| > |C_max| then
9:          C_max = C
10:      end if
11:   end for
12: end for
13: return C_max
```

Depending on what subroutine we apply to find a maximum clique C_{prm} on the permutation graphs induced by permutation segments, the overall running time of the algorithm would be $O(n^2(n+m))$ or $O(n^3 \log n)$.

3.6.2 The INDEPENDENT SET problem

We already encountered the notion of a line segment q being to the left or to the right of a permutation line segment p in the proof of Lemma 3.5.2. For simplicity, we use the notation $q < p$ and $q > p$. Clearly, two permutation line segments p_1 and p_2 do not intersect if and only if $p_1 < p_2$ or $p_1 > p_2$. We say that a line segment q is *between* two non-intersecting permutation segments p_1 and p_2, if $p_1 < q < p_2$ or $p_2 < q < p_1$.

Suppose G is an IP-SEG graph with a given model M_G. Let I be a maximum independent set in G and I_{prm} and I_{int} be the sets of permutation and interval segments, respectively, that form I. The permutation segments in I_{prm} must form a sequence $\{p_i\}$ such that each p_{i+1} is to the right of p_i.

Let p_{n-1} and p_n be the next to last and last permutation segments in the sequence $\{p_i\}$, respectively, as shown in Figure 3.15. Let $G'(p_{n-1})$ be the set of all segments, interval and permutation,

that are to the left of p_{n-1}. Let $G(p_{n-1}) = G'(p_{n-1}) \cup \{p_{n-1}\}$ and let $I(p_{n-1}) = I \cap G(p_{n-1})$. It is easy to see that if I is of maximum size in G, then $I(p_{n-1})$ must be an independent set of maximum size in the induced subgraph $G(p_{n-1})$ of G.

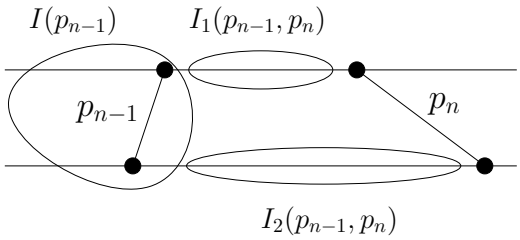


Figure 3.15: An IP-SEG representation of an independent set with at least two permutation segments.

Denote by $G_i(p_{n-1}, p_n)$ the set of all interval segments along $\mathscr{L}_i$ that are between p_{n-1} and p_n and let $I_i(p_{n-1}, p_n) = I \cap G_i(p_{n-1}, p_n)$. It is again easy to see that if I is of maximum size in G, then $I_i(p_{n-1}, p_n)$ must be an independent set of maximum size in the induced subgraph $G_i(p_{n-1}, p_n)$ of G.

While finding $I(p_{n-1})$ amounts to solving the original problem of finding I, we do know that the subgraph $G_i(p_{n-1}, p_n)$ is an interval graph and therefore, we can find $I_i(p_{n-1}, p_n)$ by applying an existing algorithm for finding an independent set of maximum size on interval graphs. This leads to a simple dynamic programming algorithm for the independent set problem in which for each permutation segment p we keep track of the largest independent set $I(p)$ formed by p along with segments that p is to the right of [67]. Begin by obtaining a left-to-right topological ordering T of the set of all permutation segments using the *to the right of* relation. Then, process segments in T in a left-to-right order. If a segment p is not to the right of any of the already processed segments from T, then $I(p)$ is simply the union of p and the largest independent sets on $\mathscr{L}_1$ and $\mathscr{L}_2$ formed by interval segments that p is to the right of. Otherwise, we also need to consider the sets $I(p') \cup I_2(p', p) \cup I_2(p', p)$ for each permutation segment p' that p is to the right of. We also need to account for the possibility that the largest independent set of G consists only of interval segments in M_G. For this, we simply need to combine the largest independent set of the interval subgraph of G induced by interval segments lying on $\mathscr{L}_1$ with the corresponding independent set on $\mathscr{L}_2$. Finally, it is possible that a largest independent set of G contains interval segments that lie to the right of p_n. We deal with this by introducing a new permutation segment p^* to M_G that lies to the right of all

other line segments. In the above notation, finding a maximum independent set of I of G amounts to finding $I(p^*) - \{p^*\}$ in (the updated version of) M_G.

Algorithm 2 INDEPENDENT SET on IP-SEG graphs (given the model)

IndependentSet(G, M_G)

```
1:  p*: a new permutation segment that is to the right of all segments in M_G
2:  M_G = M_G ∪ {p*}
3:  T: a topological ordering of the permutation segments of M_G
4:  for p in T do
5:     I_1(p): largest independent set of interval segments s on L_1 such that s < p
6:     I_2(p): largest independent set of interval segments s on L_2 such that s < p
7:     I(p) = I_1(p) ∪ I_2(p) ∪ {p}
8:     for p' in T such that p' < p do
9:        I*(p) = I(p') ∪ I_1(p', p) ∪ I_2(p', p) ∪ {p}
10:       if |I(p)| < |I*(p)| then
11:          I(p) = I*(p)
12:       end if
13:    end for
14: end for
15: return I(p*) − {p*}
```

Since we are given the model and thus we have the interval segments in sorted order, we can find each of the independent sets of interval subgraphs in the algorithm in $O(n)$ time. This leads to an overall running time of $O(n^3)$.

3.6.3 The LONGEST CHORDLESS CYCLE problem

Another problem that is NP-hard on general graphs, but can be solved in polynomial time on IP-SEG graphs when given the model, is the *longest chordless cycle* (LCC) problem. In the literature, this problem is also known as the *longest induced cycle* problem and it remains NP-complete on bipartite graphs [45]. Further, much like the CLIQUE problem, the LCC problem on general graphs is NP-hard to approximate within $n^{1-\varepsilon}$, for any $\varepsilon > 0$ [70]. However, due to the fact that the set of possible IP-SEG configurations of C_n is limited (see Figure 3.7), we can solve the LCC problem on IP-SEG graphs in polynomial time by going through these configurations. Note that this approach is not based on exhaustive enumeration as IP-SEG graphs can have exponentially many chordless cycles. For example, the graph on $2n+2$ vertices whose IP-SEG model is shown in Figure 3.16 has 2^n different chordless cycles of length $n+2$.

Nonetheless, we know that chordless cycles can only use 2-4 permutation segments in an IP-

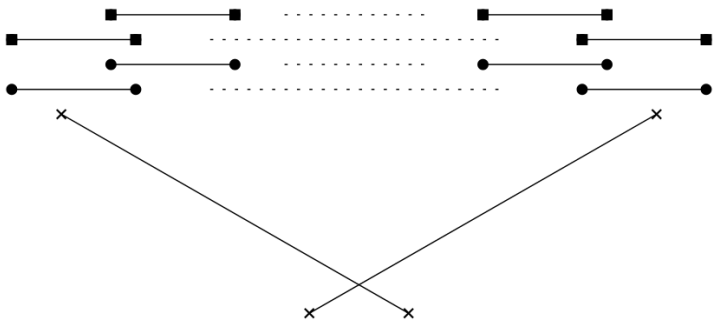


Figure 3.16: A simple IP-SEG model M_G of a graph G on $2n+2$ vertices. Each of G's chordless cycles is of length $n+2$ and consists of 2 permutation segments and n interval segments in M_G. Notice that each interval segment i_1 has a true twin i_2. Substituting i_1 with i_2 in any chordless cycle that i_2 is part of, leads to different chordless cycle.

SEG model. For a given subset of 2-4 permutation segments P, we can easily check whether they can correspond to a chordless cycle configuration (i.e. whether they can form one valid permutation arc or two disjoint valid permutation arcs). Suppose they can and that we want to find a longest chordless cycle that includes P. This can be reduced to the problem of finding a longest interval arc whose interval segments fall within a certain range. We illustrate the approach by looking at one particular configuration shown in Figure 3.17.

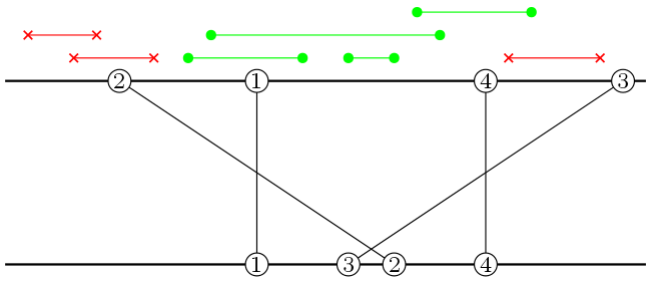


Figure 3.17: A 4-segment permutation arc $P = (p_1, p_2, p_3, p_4)$ that could potentially be combined with an interval arc I to form a chordless cycle. The interval segments in green could partake in such an interval arc I, but those in red cannot.

The set P consists of 4 permutation segments. The only way to incorporate P into an IP-SEG model of a chordless cycle C is to combine it with an interval arc I along the line $\mathscr{L}_1$. A valid interval arc I can consist only of interval segments that lie entirely between $t(p_2)$ and $t(p_3)$ and contain at least part of the interval $[t(p_1), t(p_4)]$ - these are the green segments in Figure 3.17 and we refer to them as $I_1(P)$. Thus, to find a longest chordless cycle C containing P it suffices to find a longest interval arc I in $I_1(P)$ that starts in $(t(p_2), t(p_1)]$ and ends in $[t(p_4), t(p_3))$.

Recall that an interval arc corresponds to a chordless path. Further, the problem of finding a longest chordless path in a vertex-weighted graph (WLCP) is known to be solvable in polynomial time on the class of interval graphs [47, 63]. We cannot apply such algorithms to the interval graph induced by $I_1(P)$, as a longest chordless path of interval segments in $I_1(P)$ is not guaranteed to intersect $t(p_1)$ and $t(p_4)$ and may therefore not constitute a valid interval arc. However, we can find a longest valid interval arc by transforming our problem to the following instance of the WLCP problem. Let $G_1(P)$ be the subgraph induced by vertices corresponding to the set of segments $I_1(P) \cup \{p_1, p_4\}$. It is easy to see that $G_1(P)$ must be an interval graph. Further, assign weights of n to the vertices of p_1 and p_4 and weights of 1 to all other vertices of $G_1(P)$. Then, $I_1(P)$ contains a valid interval arc of length k connecting p_1 with p_4 if and only if $G_1(P)$ contains a chordless path with total weight of $2n+k$.

Analogous reductions to instances of the WLCP problem on interval graphs can be done for each of the other IP-SEG configurations of chordless cycles. When exploring configurations with 2 interval arcs, we will actually need to solve 2 instances of the WLCP problem. Each of these reductions can be performed in $O(n)$ time when the model of the IP-SEG graph is given. Solving each instance of the WLCP problem can be done in $O(n^3)$ time [63]. Overall, we will need to consider up to $O(n^4)$ subsets of 2-4 permutation segments P, leading to a total running time of $O(n^7)$. We summarize the full algorithm for the LCC problem on IP-SEG graphs, when the model is given, below.

Note that Algorithm 3 only explores chordless cycle configurations that contain both interval and permutation segments. The LCC of a given IP-SEG graph, may be of length 3 or 4, meaning it could be composed of segments of the same type. For completeness, if the LCC found by Algorithm 3 is of length less than 4, we will need to identify if the subgraph induced by permutation segments only contains a C_4. If no chordless cycle was found at this point, we will also need to see if the subgraph induced by interval segments only contains a triangle. These additional checks, however, do not affect the overall asymptotic running time bound.

3.7 First steps towards recognition

In this section we tackle the recognition problem of IP-SEG graphs and offer some partial results. A natural place to start is to look at simpler subclasses of IP-SEG graphs like trees. We say that a

Algorithm 3 LONGEST CHORDLESS CYCLE on IP-SEG graphs (given the model)

LongestChordlessCycle(G, M_G)

$LCC = \emptyset$
$\mathscr{P}$: k-element subsets of permutation segments in M_G, with $2 \leq k \leq 4$
for $P \in \mathscr{P}$ **do**
  if P *can be part of a valid chordless cycle configuration* $\mathscr{C}$ **then**
    for $i \in \{1,2\}$ **do**
      $I_i(P)$: interval segments on $\mathscr{L}_i$ that could partake in a valid interval arc in $\mathscr{C}$
      if $I_i(P) \neq \emptyset$ **then** # $\mathscr{C}$ may or may not admit an interval arc on $\mathscr{L}_i$
        p,q: permutation segments of P delimiting valid interval arcs on $\mathscr{L}_i$ in $\mathscr{C}$
        $w : I_i(P) \cup \{p,q\} \longrightarrow \{1,n\}$: weight function s.t. $w(p) = w(q) = n$ and $w(x) = 1, \forall x \notin \{p,q\}$
        $G_i(P)$: interval graph induced by $I_i(P) \cup \{p,q\}$ and weighted by w
        $MIA_i(P) = WLCP(G_i(P)) - \{p,q\}$ # find a longest interval arc on $\mathscr{L}_i$
      else
        $MIA_i(P) = \emptyset$
      end if
    end for
    if $|P \cup MIA_1(P) \cup MIA_2(P)| > |LCC|$ **then**
      $LCC = P \cup MIA_1(P) \cup MIA_2(P)$
    end if
  end if
end for
return LCC

tree T is a *caterpillar* if the removal of all pendant vertices (leaves) of T results in a path P. We call P the *central path* of T. We say that a subtree S of a tree T is *attached* to a vertex $v \in T$ if $v \notin S$ and S is a connected component of $T - v$. Note that if a subtree S of a tree T is attached to v, then v must have exactly one neighbor in S.

Trees in the class of interval graphs are exactly caterpillars [36]. The same is true for trees in the class of permutation graphs [3, 48]. While IP-SEG graphs can contain trees that are not caterpillars, we will show that IP-SEG trees can be characterized as a fairly simple generalization of caterpillars.

Let T be a an IP-SEG graph that is a tree and let $M(T)$ be an IP-SEG model of T. Let Q be the set of permutation segments in $M(T)$. First, suppose that the subgraph $T[Q]$ induced by vertices corresponding to permutation segments is connected, i.e. is a subtree. Then $T[Q]$ must be a caterpillar. Let $q \in Q$ be an arbitrary permutation segment and let $S(q)$ be the set of vertices that can be reached from q without going through any vertices of $Q - \{q\}$. Since T is a tree and thus acyclic, $S(q)$ must consist of interval segments only. Moreover, for each $i \in \{1,2\}$, the set $S_i(q) \subseteq S(q)$ of

interval segments lying on line $\mathscr{L}_i$ must be either empty or form a caterpillar that is attached to q. Our choice of q was arbitrary, meaning that any $q \in Q$ can have up to two caterpillars formed by interval segments attached to it, regardless of whether q is part of the central path $C(T[Q])$ of $T[Q]$ or q is a leaf node of $T[Q]$. In other words, we can think of T as containing a *central caterpillar* $T[Q]$ such that each $q \in T[Q]$ may have up to 2 *peripheral caterpillars* attached to it. Note that different IP-SEG models of T may lead to different central and peripheral caterpillars, i.e. an IP-SEG tree may have multiple subtrees that could play the role of a central caterpillar.

Now let us consider the case when $T[Q]$ is not connected. Then, $T[Q]$ must consist of a set of k disjoint caterpillars that have a unique left-to-right ordering $\mathscr{O} = T[Q_1] < T[Q_2] < \cdots < T[Q_k]$ in $M(T)$. Let t_1 and b_1 be the segments in $T[Q_1]$ with the rightmost top and bottom endpoints, respectively. Note that $t_1 \cap b_1 \neq \emptyset$ and further, when $T[Q_1]$ consists of a single segment, we have that $t_1 = b_1$. Let T_1 be the set of interval segments on $\mathscr{L}_1$ that can be reached from t_1, using only interval segments. Similarly, let B_1 be the set of interval segments on $\mathscr{L}_2$ that can be reached from b_1, using only interval segments. Since T is both connected and acyclic, exactly one of T_1 and B_1 must intersect with exactly one permutation segment in $T[Q_2]$. Without loss of generality, we may assume this to be T_1. It is possible that T_1 intersects with permutation segments from other caterpillars $T[Q_j]$ for $3 \leq j \leq J$, but only if $T[Q_2], T[Q_3], \ldots, T[Q_{J-1}]$ consist of exactly one permutation segment each.

Suppose that $k = J$ and let $I_{1,k}$ be the unique path of interval segments on $\mathscr{L}_1$ connecting t_1 with the permutation segment $q_k \in T[Q_k]$ that T_1 intersects. Let $C(Q_j)$ denote the central path of the caterpillar $T[Q_j]$. Then, t_1 is either an end vertex of $C(Q_1)$ or a leaf vertex pendant to an end vertex of $C(Q_1)$. The same can be said about q_k with respect to $C(Q_k)$. Because of this, $T' = T[I_{1,k} \cup Q_1 \cup Q_2 \cup \cdots \cup Q_k]$ is a caterpillar with a central path $C(Q_1) \cup I_{1,k} \cup C(Q_k)$. Moreover, T has T' as a central caterpillar and each of the vertices of T' has at most 2 peripheral caterpillars attached to it. It is easy to see how the argument can be extended inductively to trees T where $k > J$. This leads us to the following characterization of IP-SEG trees.

Lemma 3.7.1. *A tree T is an IP-SEG graph if and only if it satisfies the property $\mathscr{P}$: T can be decomposed into one central caterpillar C and a set of peripheral caterpillars $\mathscr{C}$ such that each vertex $v \in C$ has at most two caterpillars of $\mathscr{C}$ attached to it.*

Proof. We have already shown in the above discussion that any IP-SEG tree must satisfy the property $\mathscr{P}$. Going in the opposite direction, suppose that T is a tree satisfying the property $\mathscr{P}$. We can construct an IP-SEG model of T as follows. The central caterpillar C is a permutation graph and therefore, we can construct a model of C consisting only of permutation segments using any recognition algorithm for permutation graphs that outputs a model as a by-product. Since any caterpillar is also an interval graph, we can create an interval model for each of the peripheral caterpillars in an analogous way. Finally, for each vertex $v \in C$ with a permutation segment $p(v)$:

- if v has no caterpillars attached to it, do nothing;
- if v has one caterpillar C_1 attached to it, scale the interval model of C_1 and place it along $\mathscr{L}_1$, while making sure that the appropriate segment of C_1 contains $t(p(v))$;
- if v has two caterpillars C_1 and C_2 attached to it, scale the interval models of C_1 and C_2 and place them along $\mathscr{L}_1$ and $\mathscr{L}_2$, so that the appropriate segments of C_1 and C_2 contain $t(p(v))$ and $b(p(v))$, respectively.

□

We can use the above characterization of IP-SEG trees to design a recognition algorithm. For any pair of vertices u, v from a tree T, there is a unique $u - v$ path in T. Thus, if T were an IP-SEG tree, then the central path P_C of its central caterpillar will be uniquely determined by the endpoints of P_C (which may coincide when P_C is trivial). As a consequence, when trying to determine if T is an IP-SEG tree, we only need to consider $O(n^2)$ paths as candidates for P_C. The challenge is that we do not know beforehand what caterpillars contained in T and having P_C as a central path could play the role of a central caterpillar. Moreover, we may have an exponential number of such caterpillars since there could be exponentially many subsets of vertices of T that are at distance 1 from P_C which we could add as leaves of the central caterpillar. However, we can resolve this issue by looking at a specific kind of candidate central caterpillars that are easier to construct and must exist in any tree that is IP-SEG.

Suppose that T is an IP-SEG tree with a model $M(T)$ and let q be a permutation segment in $M(T)$ that is part of the central path P_C of T's central caterpillar. Let q^* be the unique vertex of $S_i(q)$ adjacent to q, where $S_i(q)$ is a nonempty caterpillar attached to q. Deleting a vertex of degree at most

2 from a caterpillar can produce an empty set, one caterpillar, or two caterpillars. This means that, if q^* has at most 2 neighbors in $S_i(q)$, we can modify $M(T)$ by turning q^* into a permutation segment and re-positioning the interval segments of the resulting (at most two) caterpillars. Figure 3.18 illustrates the transformation which leads to a new IP-SEG model of T. This implies that we can treat q^* as a leaf of the central caterpillar, with at most two caterpillars attached to it.

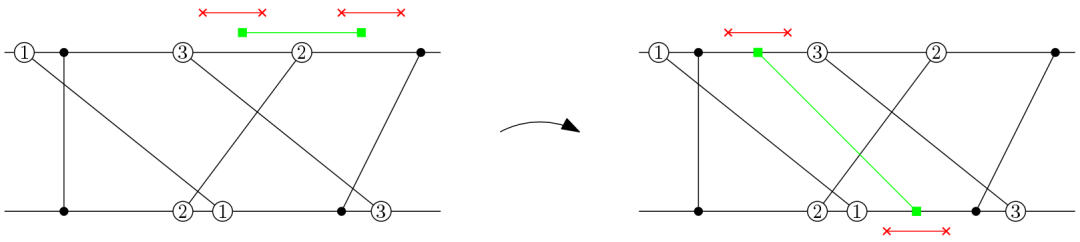


Figure 3.18: On the left, we see an IP-SEG tree with a central caterpillar formed by permutation segments only and having a central path $(1,2,3)$. 2 has a caterpillar attached to it, such that the attaching interval segment (green squares) has two neighbors (red crosses) in the attached caterpillar. On the right, we see how the attaching interval segment can be transformed to a permutation segment with two trivial caterpillars attached to it.

Applying the above transformation for all eligible caterpillars attached to a permutation segment of P_C leads to what we call a *maximal* central caterpillar. Further, any path that is a central path of at least one central caterpillar, is a central path of exactly one maximal caterpillar. Therefore, for a given tree T and a path P in T, we can check if T is an IP-SEG graph with P as the central path of some maximal caterpillar of G as follows. For any vertex $v \in P$ examine the subtrees S of T attached to v ($S_v \cap P = \emptyset$). For each such subtree S check if $S - \{u\}$ is a union of at most two caterpillars, where u is the vertex of S adjacent to v. If the answer is negative, check if S is a caterpillar itself. If S is not a caterpillar, then P cannot be the central path of a maximal caterpillar of T. Otherwise, mark S as a *special* caterpillar. If we are able to complete the process without encountering a vertex $v \in P_C$ with more than 2 special caterpillars attached to it, then G is an IP-SEG tree, otherwise it is not. We summarize the full recognition algorithm below.

The recognition algorithm for IP-SEG trees first checks in $O(n)$ time whether the graph is a tree to begin with. If the input graph is a tree, the algorithm then goes through $O(n^2)$ pairs of vertices. For a given pair u, v, the unique $u-v$ path P and all the subtrees attached to vertices of that path can all together be computer in $O(n)$ time. The potentially bigger portion of work is done in determining whether P can serve as the central path of a central caterpillar. The checks performed

Algorithm 4 Determine if a graph G is an IP-SEG tree

IsIP-SEGTree(G)

if *G is not a tree* **then**
  return *False*
end if
$\mathscr{P}$: $\{(u,v) \mid u,v \in V(G)\}$ # $u = v$ is also also accounted for
for $(u,v) \in \mathscr{P}$ **do**
  P: the unique $u-v$ path in G
  for $q \in P$ **do**
    $\mathscr{S}_q$: subtrees of G attached to q but disjoint from P
    $special_q = 0$ # number of special caterpillars attached to q
    for $S_q \in \mathscr{S}_q$ **do**
      q^*: the neighbor of q in S_q
      if *$S - q^*$ is not a union of at most 2 caterpillars* **then**
        if *S_q is a caterpillar* **then**
          $special_q = special_q + 1$
        else
          return *False*
        end if
      end if
    end for
    if $special_q > 2$ **then**
      return *False*
    end if
  end for
end for
return *True*

on each subtree S attached to a vertex of P are linear in the size of S - they are based on identifying connected subtrees of a forest and determining if a given tree is a caterpillar. Since attached subtrees are attached to exactly one vertex of P, the overall work that we end up doing on these checks for all subtrees is $O(n)$. Therefore, the full recognition algorithm runs in $O(n^3)$ time and it can easily be extended to produce an actual IP-SEG model when the input graph is an IP-SEG tree. We conclude this section with a specific example of a tree not in the class of IP-SEG graphs, shown in Figure 3.19.

3.8 Additional geometric intersection models generalizing those of interval and permutation graphs

One way to look at IP-SEG graph models, in relation to interval graph models, is as a relaxation that allows endpoints of line segments in the plane to lie on two parallel straight lines in the plane as opposed to just one. This viewpoint opens up the possibility of considering a parametric general-

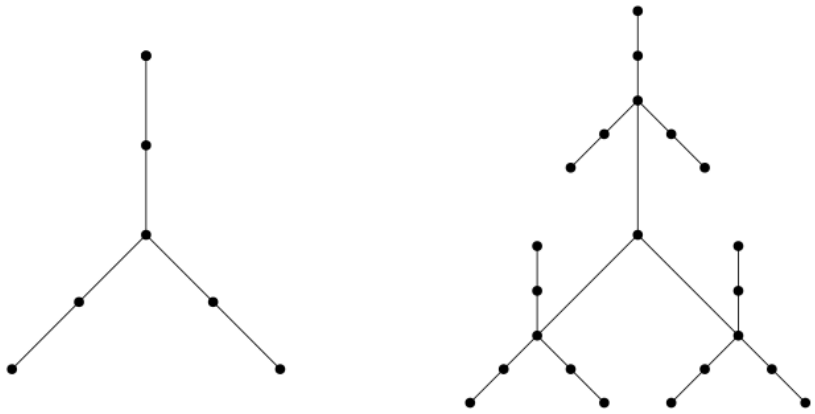


Figure 3.19: The forbidden subtree T_2 of caterpillar trees is shown on the left. T_2 is an IP-SEG tree. However, the tree shown on the right is not IP-SEG. Whatever path P we choose, at least one subtree S attached to a vertex v of P will always contain a copy of T_2 and will thus not be a caterpillar. In addition, u is the vertex in S that v is adjacent to, $S-u$ will either contain a copy of T_2 or it will consist of more than 2 connected subtrees.

ization of interval graph models, where the endpoints of line segments can be anchored at k parallel straight lines in the plane. We will call this type of model *k-IP-SEG* and the class of graphs having such a model *k-IP-SEG graphs*. Clearly, 1-IP-SEG graphs are just interval graphs and for $k \geq 2$, k-IP-SEG graphs also generalize permutation graphs.

A similar parametric generalization, which we refer to as *k-permutation*, has been considered by Middendorf and Pfeiffer in [86]. As the name suggests, in this generalization the two endpoints of a line segment cannot lie on the same parallel straight line. Clearly, k-permutation graphs are contained in the class of k-IP-SEG graphs. Because 2-permutation graphs are exactly the class of permutation graphs, the containment is proper for $k = 2$. On the other hand, it is easy to see how for any set of n arbitrarily-oriented line segments in the plane, we can draw at most $2n$ distinct parallel straight lines so that each endpoint of a line segment lies on one of the parallel lines. Thus, if we allow k to be unbounded, both parametric generalizations coincide with the general class of SEG graphs. An interesting open question is whether the containment k-permutation $\subseteq$ k-IP-SEG remains proper for all bounded values of k.

Middendorf and Pfeiffer show that given the model, the CLIQUE problem on k-permutation graphs, when k is bounded, can be solved in polynomial time. A clique in an k-IP-SEG graph with an associated model M_G can only contain interval segments coming from one parallel line in M_G. Therefore, our algorithm for the CLIQUE problem on IP-SEG graphs (see Algorithm 1) can be

naturally extended to the class of k-IP-SEG graphs, when k is bounded.

Another approach towards generalizing the geometric intersection models of interval and permutation graphs has been to allow the endpoints of segments in the plane to lie on curves other than parallel lines. In particular, circle graphs are the intersection graph of chords in a circle (line segments whose endpoints lie on a circle) and they generalize permutation graphs. Circular arc graphs are a generalization of interval graphs where instead of intersecting line segments on a straight line we have intersecting circular arcs lying on a circle. Both circle and circular arc graphs are well-studied in the literature and efficient algorithms for a number of optimization problems, including CLIQUE and INDEPENDENT SET, are known [104]. A natural extension of the two classes would be to allow for both chords and circular arcs within the same circle. Such a model clearly generalizes the IP-SEG model and it seems quite possible that similar approaches to the CLIQUE and INDEPENDENT SET problems could work on the new class of graphs.

3.9 Conclusion

In this chapter we introduced the new IP-SEG geometric intersection model simultaneously generalizing the respective models of interval and permutation graphs. We showed that the resulting classes of simple IP-SEG and IP-SEG graphs have an implicit representation. In addition, we saw that unlike earlier generalizations such as simple triangle and trapezoid graphs, these classes are not contained in the class of perfect graphs. Nonetheless, we are somewhat limited in how we can represent a chordless cycle using an IP-SEG model, which leads to some forbidden subgraphs for the two classes. Further, we demonstrated that all IP-SEG graphs have at least one vertex v whose neighborhood $N(v)$ not only cannot contain a large chordless cycle but has to induce a permutation graph. We presented polynomial algorithms for the CLIQUE, INDEPENDENT SET, and LONGEST CHORDLESS CYCLE problems on IP-SEG graphs when the model is given as part of the input. Note that each of these algorithms can be extended to solve the weighted version of the corresponding problem. We also tackled the recognition problem and presented a polynomial recognition algorithm for the special case of IP-SEG trees. Finally, we discussed parametric and non-parametric models further generalizing that of IP-SEG graphs and how some of the algorithms designed for IP-SEG graphs could be extended to work on the more general classes of graphs.

The recognition problem on the full class of IP-SEG graphs, as well as the related problem of

reconstructing an IP-SEG model of an IP-SEG graph, are natural questions that remain open. There are three lines of approach, two of which involve finding alternative characterizations of IP-SEG graphs, that appear most promising. First, given that interval and permutation graphs have nice vertex ordering characterizations and a similar result has been recently obtained for one of their generalizations - simple triangle graphs - by Takaoka [107], it would be worth exploring if such a characterization can be found for simple IP-SEG or IP-SEG graphs. A second approach would be to identify further neighborhood properties of vertices in IP-SEG graphs with the goal of expanding the list of forbidden subgraphs to a forbidden induced subgraph characterization. A third approach would be to build on our result for the recognition of IP-SEG trees and consider subclasses of IP-SEG graphs by incrementally reducing the restrictions on the kinds of cycles we allow. Potential starting points include the recognition of IP-SEG *cactus* graphs (graphs in which two cycles can share at most one vertex) [15] and IP-SEG graphs that contain only *atomic* cycles (chordless cycles C such that for each pair of vertices u, v in C, one of the shortest $u - v$ paths is fully contained in C) [46].

Future work should also be done on studying other optimization problems on the class of IP-SEG graphs. A good candidate would be COLORING, given the simple algorithms for this problem on interval and permutation graphs arising naturally from their geometric intersection models. In addition, given our result on the LONGEST CHORDLESS CYCLE problem, it could be worthwhile exploring related problems like HAMILTON CYCLE and its generalization LONGEST CYCLE, where cycles of interest do not have to be chordless. In the following chapter we will see a robust algorithm for the CLIQUE problem on IP-SEG graphs based on a vertex elimination ordering. It would be interesting to know if we can design robust algorithms for optimization problems other than CLIQUE. Finally, future study could involve better understanding the generalizations of IP-SEG graphs discussed in Section 3.8 and extending additional algorithmic results from IP-SEG graphs to the IP-SEG generalizations.

CHAPTER IV

Vertex Orderings

4.1 Introduction

There are different methods for defining or characterizing graph classes. We have already seen intersection models characterizing a number of classes, including interval, permutation, and IP-SEG graphs. Other characterization methods include containment models, forbidden subgraphs, intersection of properties, vertex orderings, and edge orderings [104]. Some well-studied classes like permutation graphs have known characterizations of multiple types, including:

- **Intersection model** [104]: Line segments going between two parallel lines;
- **Containment model** [15, 35, 52, 104]: Intervals on the real line;
- **Forbidden induced subgraphs** [28, 44]: No induced subgraphs from an infinite set of graphs of various kinds, that includes odd-holes;
- **Intersection of properties** [35, 104]: Graphs that are both comparability and co-comparability;
- **Vertex ordering** [15, 104]: An ordering of the vertices $v_1, v_2, \ldots, v_n$ that does not have any triples (v_i, v_j, v_k), $1 \leq i < j < k \leq n$, such that:
  1. $v_i v_k$ is an edge, but $v_i v_j$ and $v_j v_k$ are not edges, or
  2. $v_i v_k$ is not an edge, but $v_i v_j$ and $v_j v_k$ are edges.

In this chapter, our focus will be on vertex orderings. A *vertex ordering characterization* (VOC) of a class $\mathscr{G}$ usually has the following form: a graph G is a member of the class $\mathscr{G}$ if and only if G has a vertex ordering satisfying one or more properties [55, 107]. These characterizing vertex orderings can often be a powerful tool in developing efficient algorithms for various problems on the corresponding graph class. Such algorithms process these vertex orderings from left to right or right to left, looking at local properties of a single vertex at a time [15]. The vertex ordering for permutation graphs shown above allows for a linear dynamic programming algorithm that solves the clique problem [104]. Similarly, we can solve the graph coloring problem on chordal graphs in

linear time by stepping through a reversed perfect elimination ordering [15]. Vertex orderings can help us develop efficient algorithms for optimization problems even when they do not constitute a VOC of a class of interest. In fact, there are instances in which such vertex orderings allow us to design robust algorithms for an optimization problem on the class.

In Section 4.2 we introduce some notation and basic definitions that will be used throughout the chapter. In Section 4.3 we explore several types of vertex orderings, differing in the way the defining property is specified. These include elimination orderings and vertex orderings that avoid certain forbidden ordered subgraphs. We also explore the complexity of the related problems of class recognition and representation construction on different vertex ordering types. In Section 4.4 we look at when and how different types of vertex ordering characterizations are useful in designing efficient algorithms for important optimization problems. We give the first robust polynomial algorithm for the CLIQUE problem on IP-SEG graphs and discuss a generalization of the approach to other classes of graphs. In Section 4.5 we focus on multiple-property vertex ordering characterizations as a method of combining graph class properties. We explore when this approach leads to a more restricted graph class than simply taking the intersection of the corresponding graph classes. Further, we discuss situations where certain optimization problems may become tractable on the more restricted class.

4.2 Preliminaries

If a graph G has a linear ordering $<$ of its vertices associated with it, we write $G = (V, E, <)$ and we call G an *ordered graph*.

An ordered graph $G' = (V', E', <_{G'})$ is an *induced ordered subgraph* of an ordered graph $G = (V, E, <_G)$ if G' is an induced subgraph of G and $<_{G'}$ is a suborder of $<_G$.

We call G^R the *reversal* of an ordered graph G if $V(G^R) = V(G)$ and $E(G^R) = E(G)$, but $<_{G^R}$ is obtained by reversing the ordering $<_G$. The *complement* of an ordered graph $G = (V, E, <_G)$, denoted by G^C, is the unordered complement $\overline{G}$ of G combined with the linear ordering $<_G$.

We now formalize the two main methods of graph class characterization that we will be referring to in this chapter.

Definition 4.2.1. We say that a graph class $\mathscr{G}$ has an *intersection of properties characterization (IPC)* in terms of graph classes $\mathscr{G}_1, \mathscr{G}_2, \ldots,$ and $\mathscr{G}_n$, if $\mathscr{G} = \mathscr{G}_1 \cap \mathscr{G}_2 \cap \cdots \cap \mathscr{G}_n$.

The reason we call the above characterization method “intersection of properties", instead of just “intersection of graph classes", is that the intersecting graph classes are themselves characterized by possessing certain properties. Any graph that is a member of the new graph class satisfies each of the properties characterizing the intersecting graph classes.

Definition 4.2.2. Let $\mathscr{G}$ be a class of graphs and let $\mathscr{P} = \{P_1, P_2, \ldots, P_n\}$ be a set of properties. We say that $\mathscr{G}$ admits a *vertex ordering representation (VOR)* in terms of properties $\mathscr{P}$ if for any graph $G \in \mathscr{G}$, there exists a linear ordering of the vertices $V(G)$ that satisfies the properties $\mathscr{P}$. We say that $\mathscr{G}$ has a *vertex ordering characterization (VOC)* in terms of properties $\mathscr{P}$, if for any graph G, $G \in \mathscr{G}$ if and only if there exists a linear ordering of the vertices $V(G)$ that satisfies the properties $\mathscr{P}$.

Note that a VOR or a VOC, as defined above, requires a single vertex ordering that satisfies all of the properties $\mathscr{P}$ simultaneously. The importance of this distinction will become more apparent in Section 4.5 when we look at VOC as a potentially more restrictive approach than IPC for combining multiple vertex ordering properties. In addition, the notion of a VOR will be particularly useful in the context of graph classes for which we do not yet have a nice VOC, but can identify a VOR that helps us solve a particular optimization problem.

4.3 Vertex ordering types

Vertex ordering properties can take different forms, but they can be broadly fitted into two main types. The first type includes properties that a vertex v needs to satisfy with respect to vertices that come after it in a vertex ordering. While it is generally important which vertices come after v in the vertex ordering, the property that v needs to satisfy is usually specified in a way that does not put any restrictions on the relative ordering of said vertices. The vertex ordering characterizations induced by such properties are most commonly referred to as *elimination orderings*. The second type of properties are usually specified in terms of a set of patterns of vertices and edges that a valid vertex ordering needs to avoid. These forbidden patterns are most commonly expressed as a set of small ordered graphs.

4.3.1 Elimination orderings

We use the following formalization of an *elimination vertex ordering satisfying a property* taken from [15].

Definition 4.3.1. The ordering $(v_1, v_2, \ldots, v_n)$ of the vertex set V of a graph G is an elimination ordering satisfying property P if for all $i \in \{1, \ldots, n\}$, the vertex v_i has property P in the graph $G_i = G[v_i, v_{i+1}, \ldots, v_n]$.

One very important example of an elimination ordering characterizes the class of *chordal* graphs - the graphs in which every cycle of four or more vertices has a chord.

Theorem 4.3.1. ***(Dirac [32])*** *A graph G is chordal if and only if there exists an ordering of its vertices $(v_1, v_2, \ldots, v_n)$ such that each vertex v_i is simplicial in $G_i = G[v_i, v_{i+1}, \ldots, v_n]$. We call such a vertex ordering a* ***perfect elimination ordering****.*

Note that chordal graphs are defined by a global property as the graphs that do not contain an induced chordless cycle on four or more vertices. However, the property of a vertex being simplicial in the above VOC characterization is local. This, combined with the fact that it is easy to test if a specific vertex is simplicial, leads to a straightforward greedy polynomial recognition algorithm for chordal graphs: find a simplicial vertex, eliminate it from the graph, and then repeat the same for the remaining subgraph. If at any point the algorithm fails to find a simplicial vertex, the graph is not chordal. If, however, the algorithm is able to eliminate all vertices of the graph then the graph is chordal and the elimination ordering is perfect.

Another example of a graph class defined by a global property, but having a simple and useful VOC that uses a local property is that of *distance hereditary* graphs. A graph G is *distance hereditary* if and only if for every pair of vertices x and y all induced paths between x and y have the same length. Distance hereditary graphs also have the following VOC that uses the local property of being a pendant vertex or a twin vertex.

Theorem 4.3.2. ***(Bundelt et al. [7])*** *A graph G is distance hereditary if and only if there exists an ordering of its vertices $(v_1, v_2, \ldots, v_n)$ such that each vertex v_i is pendant to, or is the twin of, a vertex in $G_i = G[v_i, v_{i+1}, \ldots, v_n]$. We call such a vertex ordering a* ***pruning sequence****.*

As was the case with chordal graphs, the above local-property VOC of distance-hereditary graphs allows for a naive greedy recognition algorithm. While there are more efficient linear time recognition algorithms for both classes [104], the fact remains that graph classes having an elimination VOC are generally easy to recognize, one approach being through the construction of the corresponding elimination ordering. We will see that the same is not always true for graph classes that only have a VOC that forbids certain ordered patterns.

4.3.2 Forbidden ordered subgraphs

The other main type of vertex ordering properties are usually stated as follows: $<$ is a valid vertex ordering of G if $<$ contains no subordering from a set S [15]. In the most common formulation due to Damaschke [27], the set S consists of ordered graphs. Such a vertex ordering becomes a representation (characterization) of a graph class, if (and only if) every graph from the graph class has a vertex ordering avoiding the ordered subgraphs from S. Two forbidden ordered subgraphs on three vertices, denoted by the abbreviations *ch* and cp, that are used in vertex ordering characterizations of a number of well known graph classes are shown in Figure 4.1.

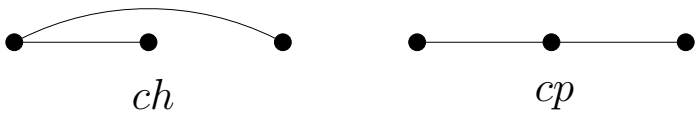


Figure 4.1: The ordered subgraphs *ch* and cp.

The following theorem states several forbidden ordered subgraph characterizations of graph classes expressed in terms of *ch* and cp, as well as their reversals or complements. Note that most of these characterizations were known in the literature prior to [27], but appeared in different contexts and were not identified as belonging to the same type of VOC's.

Theorem 4.3.3. ***(Damaschke [27])*** *For each pair $(\mathscr{G},S)$ in Table 4.1, $G \in \mathscr{G}$ if and only if there is a vertex ordering of G that contains no induced ordered subgraph from S.*

It is easy to see that perfect elimination orderings mentioned earlier are exactly the orderings that avoid *ch*. Thus, for chordal graphs, elimination orderings and forbidden ordered subgraphs are just two different ways of describing the same characterization. A similar statement can be made about threshold graphs, whose elimination ordering is such that each vertex v_i must be either isolated or neighboring all vertices in $G_i = G[v_i, v_{i+1}, \ldots, v_n]$.

Table 4.1: Classes of graphs characterized by sets of forbidden ordered subgraphs.

(1)	$\mathscr{G} =$ **chordal**	$S = \{ch\}$
(2)	$\mathscr{G} =$ **comparability**	$S = \{cp\}$
(3)	$\mathscr{G} =$ **threshold**	$S = \{ch, ch^C\}$ or $S = \{ch^R, ch^C, cp\}$
(4)	$\mathscr{G} =$ **split**	$S = \{ch, ch^{CR}\}$
(5)	$\mathscr{G} =$ **proper interval**	$S = \{ch, ch^R\}$
(6)	$\mathscr{G} =$ **interval**	$S = \{ch, cp^C\}$
(7)	$\mathscr{G} =$ **permutation**	$S = \{cp, cp^C\}$

Comparability graphs arise from partial orders and are the graphs whose edges can be oriented *transitively*, i.e. in such a way that for any triple of vertices a, b, c, if we have the directed edges $a \to b$ and $b \to c$ then we also must have the edge $a \to c$. Such a transitive orientation of the edges of a graph produces an acyclic directed graph $\vec{G}$ and it is not difficult to see that the topological orders of $\vec{G}$ are exactly the vertex orderings of G that avoid cp.

Split graphs are those whose vertices can be partitioned into a clique and an independent set. They can also be characterized as the graphs that are both chordal and co-chordal. Using this second characterization, one can easily derive the forbidden ordered subgraph characterization stated in Theorem 4.3.3. Threshold graphs are a proper subclass of split graphs and we will revisit this relationship in Section 4.5.

We have already seen that permutation graphs are defined as the intersection graphs of line segments with endpoints on two parallel lines and interval graphs as the intersection of intervals on the real line. Proper interval graphs are a subclass of interval graphs with a geometric intersection model in which no interval properly contains another.

We should note that in the more recent literature, the equivalent notion of *forbidden patterns* has been used in place of forbidden ordered subgraphs [40, 61, 107]. In its original formulation [61], a pattern on k vertices is just an ordered graph on k vertices, but specified in such a way that the vertices always have the labels $\{1, 2, \ldots, k\}$, i.e. the vertex ordering of the graph is implied by the vertex labels. Some authors, however, expand the notion of a *pattern* to include edges that may or may not be present in the corresponding ordered subgraph [40, 107]. More precisely, they define a pattern as an ordered graph in which a pair of vertices has one of three relations: plain edges, dashed edges, and non-edges. A subgraph of an ordered graph matches a pattern, if the plain edges are present, the non-edges are absent, and there is no constraint on dashed edges. Then, a graph class

$\mathscr{G}$ has a VOC in terms of a set of forbidden patterns S if each graph $G \in \mathscr{G}$ has a vertex ordering such that no ordered subgraph of G matches a pattern in S. Clearly a pattern with no dashed edges is just an ordered subgraph. The presence of a dashed edge xy implies that the given pattern encompasses both ordered graphs that have the edge xy and ordered graphs that do not have it. Thus, as in the case of interval graphs shown in Figure 4.2, patterns may lead to a more compact representation of a VOC than ordered subgraphs.

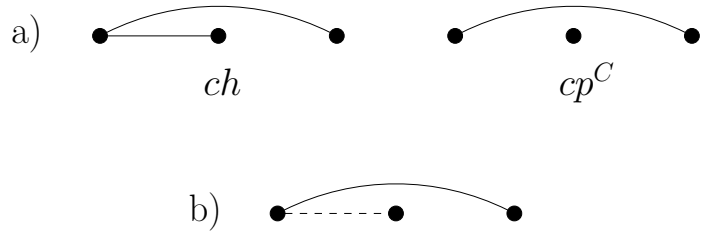


Figure 4.2: a) The two forbidden ordered subgraphs characterizing interval graphs. b) The forbidden pattern characterizing interval graphs. The dashed edge indicates that this pattern encompasses both ch and cp.

If a graph class $\mathscr{G}$ has a forbidden ordered subgraph characterization in terms of the set $S = \{s_1, s_2, \ldots, s_k\}$, we simply write $\mathscr{G} = (s_1, s_2, \ldots, s_k)$. Thus, from Theorem 4.3.3 we have that $chordal = (ch)$ and $interval = (ch, cp^C)$. We will adopt the convention of using lowercase letters for denoting forbidden ordered subgraphs, reserving the use of uppercase letters for vertex ordering properties expressed in a different way. For example, we will use (DH) to denote a vertex ordering that is the pruning sequence of a distance hereditary graph, since we do not know of a finite set of forbidden ordered subgraphs characterizing this graph class.

4.3.3 Complexity of recognition

We saw that classes like chordal and threshold graphs that have elimination ordering characterizations admit a naive greedy algorithm for constructing the special ordering which also serves as a recognition algorithm. The other graph classes mentioned in Theorem 4.3.3 do not have a known elimination ordering. Despite that, we can efficiently recognize them (in polynomial time). The algorithms are not as straightforward as greedily building an elimination ordering, but all of the graph classes in Theorem 4.3.3 can be recognized in $O(n+m)$ time, except comparability graphs for which the best known algorithms require time proportional to matrix multiplication. In addition, Hell et al. [61] have shown that all graph classes having a VOC in terms of forbidden patterns of

3 vertices can be recognized in polynomial time. Thus, a natural question to ask is whether graph classes with a VOC can always be recognized in polynomial time. The answer, assuming $P \neq NP$, is no.

Before we give the first example of a graph class with a VOC that is NP-complete to recognize, we need to define several notions taken from Chaturvedi in [20].

Definition 4.3.2. Let $\vec{G}$ be a directed acyclic graph (DAG). We say that a directed path $\vec{p} = p_1 \to p_2 \to \cdots \to p_b$ in $\vec{G}$ is **open** if the ends p_1 and p_b are not adjacent. We say that $\vec{G}$ is **β-transitive** if the number of vertices in a longest open directed path of $\vec{G}$ is equal to β. An undirected graph G is **β-orientable** if there is an acyclic orientation of its edges such that the resulting DAG is β-transitive. **β-ORIENT** is the following decision problem: Given an undirected graph G and a number k, does there exist an acyclic orientation of the edges of G such that the resulting DAG is β-transitive for some $\beta \leq k$?

The notion of β-transitivity generalizes that of transitivity and 2-orientable graphs are exactly the class of comparability graphs. Thus 2-orientable graphs can be recognized in polynomial time. Chaturvedi cites the following hardness result in [20], which is attributed to Xu.

Theorem 4.3.4. *β-ORIENT is NP-complete.*

A closer look at the actual reduction from MONOTONE NAE-3SAT to β-ORIENT presented in [20] reveals that the following stronger result also holds.

Proposition 4.3.1. *3-ORIENT is NP-complete.*

Now consider the set $S = \{b_0, b_1, b_2, b_3\}$ of ordered subgraphs shown in Figure 4.3. We claim that the graph class with a VOC in which the elements of S are forbidden, i.e. (b_0, b_1, b_2, b_3), is exactly the class of 3-orientable graphs. Indeed, if an undirected graph G is 3-orientable, it is easy to see that any topological order of the corresponding 3-transitive DAG $\vec{G}$ can serve as a vertex ordering of G that avoids all elements of S. In addition, one can show that any ordered subgraph on four or more vertices that does not have an edge between the first and the last vertex, must contain as an induced ordered subgraph one of the elements of S. Thus, if $G \in (b_0, b_1, b_2, b_3)$ and L is a vertex ordering of G that avoids the elements of S, orienting each edge xy as $x \to y$ if and only if $x <_L y$, produces a 3-transitive DAG $\vec{G}$.

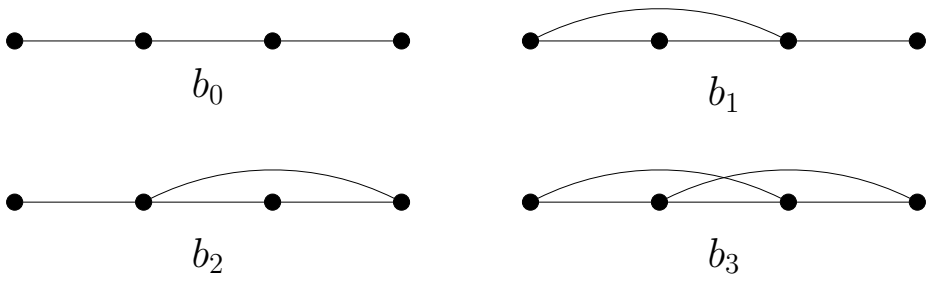


Figure 4.3: The ordered subgraphs b_0, b_1, b_2, and b_3.

Thus, the class of 3-orientable graphs has a VOC, but is NP-complete to recognize. In addition, the larger graph class (b_0) containing 3-orientable graphs is also known to be NP-complete to recognize [34]. The following result by Duffus et al. establishes the hardness of recognition for a much larger set of forbidden ordered subgraphs.

Theorem 4.3.5. ***(Duffus et al. [34])*** *Let* $\mathbf{g}$ *be a 2-connected ordered graph with*

$$<_g = (v_1, v_2, \ldots, v_k),$$

whose automorphism group contains neither of the mappings

$$(v_1, v_2, \ldots, v_k) \rightarrow (v_2, v_3, \ldots, v_k, v_1) \text{ and } (v_1, v_2, \ldots, v_k) \rightarrow (v_k, v_{k-1}, \ldots, v_2, v_1).$$

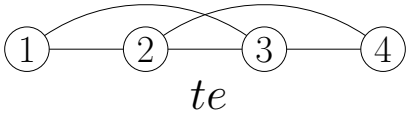
Then determining if a graph belongs to the class $\mathscr{G} = (g)$ *is NP-complete.*

Given that *almost all* graphs are 2-connected and *almost all* graphs have the trivial automorphism group consisting of only the identity mapping $(v_1, v_2, \ldots, v_k) \rightarrow (v_1, v_2, \ldots, v_k)$, Duffus et al. propose the following conjecture.

Conjecture 4.3.1. ***(Duffus et al. [34])*** *For any ordered graph g, such that g is neither complete nor an empty graph, the recognition problem for the graph class* (g) *is NP-complete if either g or its complement is 2-connected.*

This conjecture remains unresolved. One particular instance that was open until recently is the recognition problem for the class of *triangle-extendible (TE)* graphs. Consider the ordered graph te shown in Figure 4.4. A graph is TE if and only if it has a vertex ordeing that avoids te, i.e. TE $= (te)$. The class of TE graphs was introduced by Spinrad [104] as another generalization of comparability graphs. te is isomorphic to the ordered subgraph b_3 - one of the graphs used to define 3-orientable graphs. Thus, every 3-orientable graph is also triangle-extendible.

Note that te does not satisfy the assumptions of Theorem 4.3.5 - the automorphism group of te contains the mapping $(1,2,3,4) \rightarrow (4,3,2,1)$. However, it does satisfy the assumptions of Conjecture 4.3.1 - te is not complete or empty and it is 2-connected. Neogi et al. have recently shown that the recognition problem for triangle-extendible graphs is in fact NP-complete [88], offering further support for Conjecture 4.3.1. Given that both the proof of Theorem 4.3.5 and the proof of Neogi et al. use the 3-COLORING problem to establish the respective NP-completeness results, it would be interesting to see if the two can be generalized to positively resolve Conjecture 4.3.1. We will explore the class of triangle-extendible graphs, along with related subclasses and superclasses, in more detail in later sections.



te

Figure 4.4: The forbidden ordered subgraph te characterizing the class of triangle extendible graphs.

One possible approach for dealing with graph classes with VOCs, for which we do not have a polynomial recognition algorithm, is to consider subclasses on which the special vertex ordering can be constructed greedily. In other words, we want to be able to construct an elimination ordering of the vertices in such a way that when we are picking the next vertex x_i, we can be certain that x_i cannot form any violations with vertices that are yet to be eliminated, regardless of the order in which these vertices are later eliminated.

Consider a graph G belonging to the class (b_0), which we know is NP-complete to recognize. If G has a vertex v which does not appear as one of the endpoints in any induced path on four vertices in G, then G must have a vertex ordering that avoids b_0 and starts with v. Not all graphs in b_0 have such a vertex. For example, every vertex of C_5 - a cycle on 5 vertices - appears as the endpoint of two induced paths on four vertices, even though $(1,4,2,5,3)$ is a vertex ordering of C_5 that avoids b_0. Nonetheless, by only considering graphs for which we can greedily construct a b_0-free ordering, at each step eliminating the next vertex that does not start or end an induced path on four vertices, we obtain a subclass of (b_0) graphs for which we can solve the recognition problem, and recover the special vertex ordering, in polynomial time.

4.4 Optimization Problems

One of the main motivations behind the study of special graph classes is that important optimization problems, that are hard on general graphs, can become tractable. Knowing a vertex ordering, with a particular set of properties, of a graph is often very helpful in designing efficient algorithms for such problems. There are three main lines of approach that take advantage of this. One is trying to find a nice vertex ordering characterization of a graph class defined using a different model (not involving vertex orderings) and then using the properties of the characterizing vertex ordering to come up with an algorithm that solves a particular optimization problem P efficiently. The second approach involves taking a vertex ordering with properties that lead to a straightforward algorithm for a particular problem and then studying the class of graphs characterized by that vertex ordering. This often comes in the form of generalizations like 3-orientable or triangle-extendible graphs of a well-studied class like comparability graphs. Note that this mirrors our approach with IP-SEG graphs in the previous chapter where we studied geometric intersection models generalizing those of interval and permutation graphs, while trying to keep properties of the model that allowed for simple polynomial algorithms for the CLIQUE and INDEPENDENT SET problems. The third approach is a variation of the first - instead of finding a VOC of the class of interest, we try to identify a VOR of the class that can be recovered in polynomial time. Then, if we develop a polynomial algorithm that solves P by using the special vertex ordering, we are effectively solving P on a larger class of graphs. Our main contribution in this section - a robust algorithm for the CLIQUE problem on IP-SEG graphs - falls under the last category.

4.4.1 Using a VOC of a known graph class

Consider the class of chordal graphs which was originally defined as consisting of graphs that do not contain a chordless cycle of length 4 or more as an induced subgraph. Chordal graphs can also be characterized by having a perfect elimination ordering, which is used to design efficient algorithms for several problems, including CLIQUE, INDEPENDENT SET, and COLORING.

Let G be a chordal graph and let $<_G$ be a perfect elimination ordering of G. Further, let C be an arbitrary maximal clique in G and let v_i be the vertex of C having the leftmost position in $<_G$. Since v_i is simplicial in $G_i = G[v_i, v_{i+1}, \ldots, v_n]$, C is the unique largest clique with v_i as its leftmost vertex in $<_G$. Thus, an arbitrary vertex v_j can be the leftmost vertex in $<_G$ of at most one maximal clique,

implying that the number of maximal cliques of a chordal graph is at most n. Moreover, the above argument indicates a straightforward algorithm for finding a maximum clique of a chordal graph G with a given perfect elimination ordering $<_G$. Process the perfect elimination ordering from left to right and identify the vertices v_i whose neighborhood in G_i is of the largest possible size. For each such v_i, $N[v_i]$ would be a clique of maximum size.

To find the maximum independent set of a chordal graph G with a given perfect elimination ordering $<_G$, we step through $<_G = (v_1, v_2, \ldots, v_n)$ from left to right, greedily selecting a vertex v_i if none of the vertices already included in the independent set are its neighbors. To see that this greedy algorithm is correct, suppose that it makes a first mistake when selecting the k-th vertex v_{i_k}. That would mean that there is a maximum independent set I that contains the vertices $v_{i_1}, v_{i_2}, \ldots, v_{i_{k-1}}$, but there is no maximum independent set that also contains v_{i_k}. Now, the algorithm selected v_{i_k} since it did not have any neighbors in the set among $v_{i_1}, v_{i_2}, \ldots, v_{i_{k-1}}$. Furthermore, since the algorithm was greedy, all vertices after $v_{i_{k-1}}$ in the set I must appear after v_{i_k} in $<_G$. Since $<_G$ is a perfect elimination ordering, v_{i_k} can have at most one neighbor among the vertices of I appearing after v_{i_k} in $<_G$. In addition, v_{i_k} must have at least one neighbor in I, otherwise $I^+ = I + \{v_{i_k}\}$ would be an independent set properly containing I which would contradict the maximality of I. That would mean that v_{i_k} has exactly one such neighbor v^*. But then $I^* = I - \{v^*\} + \{v_{i_k}\}$ is a maximum independent set containing v_{i_k}, contradicting the assumption that selecting v_{i_k} was a mistake.

Consider the following *greedy coloring* algorithm. Let G be a graph along with an ordering of its vertices $<_G$. Suppose also that the colors are identified with the positive integers $\{1, 2, 3, \ldots\}$. Step through $<_G$ from left to right, coloring vertex v_i with the lowest-integer color not used on any already processed neighbor of v_i. On general graphs, finding an optimal coloring is NP-hard and so a greedy coloring will often be suboptimal. However, if G is a chordal graph and $<_G$ is a reversed perfect elimination ordering, the above greedy algorithm produces an optimal coloring. This stems from the fact that v_i and its already colored neighbors form a clique C. One can use this in an inductive argument to show that the color that will be assigned to v_i is exactly $|C|$. This, in turn, implies that the total number of colors used will be equal to the maximum clique size, which is a lower bound on the number of colors needed on any graph.

All of the above algorithms can be made to run in $O(n+m)$ time. In all of them, we assumed that we are given a chordal graph along with a perfect elimination ordering. We will later see

cases where we do not know how to efficiently recognize a graph class and find the special vertex ordering. Nevertheless, we can recognize chordal graphs by constructing a perfect elimination ordering in $O(n+m)$ time.

The first, constructive stage of the algorithm uses a modification of the breadth-first search (BFS) algorithm called lexicographic BFS (LexBFS). We include the description of LexBFS presented in [104]. Vertices are partitioned into sets, all vertices being in the same set initially. At each step, an arbitrary vertex x from the last set is chosen. We remove x from the graph and place it in the output list. Each set S is subdivided into neighbors and non-neighbors of x, with neighbors of x in S placed immediately after the set of non-neighbors of x in S. Standard data structures can be used to make LexBFS run in $O(n+m)$ time. The output of LexBFS is a vertex ordering. The following result due to Rose, Tarjan, and Lueker, shows the close connection between LexBFS and chordal graphs.

Theorem 4.4.1. ***[15]*** *The following two conditions are equivalent:*

(i) *G is chordal;*

(ii) *Every LexBFS vertex ordering of G is a reversed perfect elimination ordering.*

Thus, the first stage of the recognition algorithm for chordal graphs will consist of running LexBFS. In the second stage, we need to verify that the reversal of the LexBFS ordering from stage one is a perfect elimination ordering. Step through the reversal of the LexBFS ordering from left to right. Let v be the vertex considered at a given step. Find the first neighbor of v that appears after it and call this vertex w. Verify that w has edges to all neighbors of v that come after w. If this is not the case, answer that the ordering is not a perfect elimination ordering since v is not simplicial. If the verification step works at all vertices, the graph is chordal and the ordering is a perfect elimination ordering. Since the verification step for each vertex v takes time proportional to $|N(v)|$ for each vertex v, the verification stage runs in $O(n+m)$.

Having an $O(n+m)$ recognition algorithm for chordal graphs that produces a perfect elimination ordering means that we can solve the CLIQUE, INDEPENDENT SET, and COLORING problems in $O(n+m)$ time without the *promise* that a graph G is chordal and without being given a perfect elimination ordering as an additional part of the input.

We should note that the above three problems are solvable in polynomial time on the larger class of *perfect* graphs that contains chordal graphs. Perfect graphs were introduced by Berge in

the 1960s. A graph G is perfect if and only if for every induced subgraph H of G, the chromatic number of H is equal to the maximum clique size of H. Grötschel, Lovász, and Schrijver [54] used the ellipsoid method for convex programming to develop polynomial algorithms for the above three problems. However, the ellipsoid method is generally considered impractical and polynomial combinatorial algorithms for subclasses of perfect graphs, like the vertex-ordering-based algorithms for chordal graphs, remain very much of interest.

We have seen three problems that are solvable in polynomial time on perfect graphs, and in particular on chordal graphs. There are, however, a number of standard optimization problems that remain NP-hard on chordal graphs. These include the problems of finding a dominating set of minimum size and a vertex ordering of minimum bandwidth. In what follows, we will discuss the class of interval graphs which is a proper subclass of chordal graphs. In particular, we will discuss how the interval graph vertex ordering characterization that we have seen in Section 4.3 played a key role in the design of the first linear-time algorithms for the DOMINATING SET and BANDWIDTH problems on this class.

Recall that interval graphs have a very nice and simple geometric intersection model in which vertices correspond to intervals on the real line and two vertices are adjacent if and only if the corresponding intervals intersect. The extensive research on interval graphs began in the 1960s, partly motivated by an important work of Benzer [10] that dealt with the internal organization of genes and is directly related to interval graphs [104]. There are many different known characterizations of interval graphs, including the the intersection of properties characterization due to Gilmore and Hoffman [104] which states that interval graphs are the intersection of chordal and co-comparability graphs. In this section, however, we are interested in the vertex ordering characterization *interval* $= (ch, cp^C)$, i.e. interval graphs are those that have a vertex ordering that avoids the ordered subgraphs ch and cp^C shown in Figure 4.5.

Figure 4.5: The forbidden ordered subgraphs ch and cp^C characterizing interval graphs.

Suppose we are given an interval representation I of an interval graph G. In particular, let $i(v)$ be the interval corresponding to vertex v and let $l(v)$ and $r(v)$ be the left and right endpoints of $i(v)$,

respectively. We can obtain a (ch, cp^C) vertex ordering of G by ordering the vertices of G based on the right endpoints of the intervals: for any two vertices u and v, if $r(u) < r(v)$, place u before v; if $r(u) = r(v)$, use the left endpoints to break the tie, placing u before v if $l(u) < l(v)$. There are at least two works which appear to have independently discovered the (ch, cp^C) VOC of interval graphs and how to obtain it from an interval representation [77, 96].

Let G be an ordered graph with a vertex ordering $<_G$. Denote by $p(v)$ the position of vertex v in $<_G$, where the positional indices run from 1 to n. The bandwidth of G is the maximum difference $p(u) - p(v)$ across all pairs of adjacent vertices u, v. Consider the ordered graphs P_4 and C_3, as shown in Figure 4.6. P_4 has bandwidth of 1 and C_3 has bandwidth of 2.

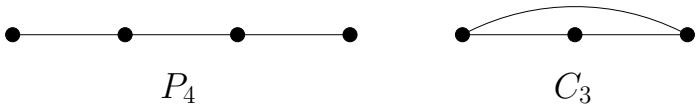


Figure 4.6: The natural ordering of the path of four vertices and the only possible ordering of a triangle.

The minimum bandwidth of an unordered graph G is the smallest bandwidth that can be achieved across all vertex orderings of G. The orderings of P_4 and C_3 shown in Figure 4.6 are the vertex orderings that achieve the minimum bandwidth, respectively. Any other vertex ordering of P_4, however, will lead to a suboptimal bandwidth of 2 or more. The BANDWIDTH problem is particularly interesting in the context of vertex ordering characterizations because it requires us to find a special vertex ordering. In particular, we can think of the class of graphs of minimum bandwidth k as having a VOC of a different type - one that is not an elimination ordering or avoids certain forbidden ordered subgraphs, but one that minimizes a certain quantity.

Kratsch [77] appears to have been the first to identify and use the (ch, cp^C) vertex ordering of interval graphs in an algorithmic context. He designed one of the first polynomial time algorithms for the decision version of the BANDIWDTH problem on interval graphs. His algorithm uses the (ch, cp^C) ordering of an interval graph as a starting point. It then processes one vertex at a time, modifying the current vertex ordering while trying to not increase the distance between already processed neighbors beyond k. At the end, it produces a vertex ordering achieving a bandwidth of at most k if such an ordering of the interval graph exists. The overall running time of this algorithm is $O(n^2)$. This then leads to a straightforward $O(n^2 \log n)$ time algorithm for the optimization version of

the problem which performs a binary search on the n possible bandwidth values using the decision version algorithm as a subroutine. It was later discovered by Mahesh et al. [83] that one of the rearrangement steps in Kratsch's algorithm was flawed. They were, however, able to correct that particular step preserving the overall structure and running time of the original algorithm.

[96] is a work that appeared several years later and used the (ch, cp^C) vertex ordering of interval graphs to solve variations of the DOMINATING SET problem. Given a graph $G = (V,E)$, $S \subseteq V$ is a *dominating* set of G if every vertex in $V - S$ has a neighbor in S. S is a *totally dominating* set if every vertex in V, including those in S, has a neighbor in S. The standard dominating set problem consists of finding a dominating set of minimum cardinality, or minimum total weight if vertices have weights. Variations of the problem specify additional properties that the dominating set needs to satisfy. In [96], Ramalingam et al. consider the properties of the set being independent, connected, or totally dominating. They then develop a unified algorithm that can be applied to each of the four weighted variations of the problem and runs in linear time. The (ch, cp^C) ordering is a key building block of the algorithm, since the decision of whether to include a vertex in the dominating set or not is based in part on the positions of its neighbors in the ordering. In addition to their algorithm being simultaneously applicable to these four variations of the DOMINATING SET problem, it was actually one of the first polynomial algorithms for weighted total domination and improved the best known bound at the time for weighted connected domination.

We should note that both the BANDWIDTH and the DOMINATING SET algorithms mentioned above relied on a recognition algorithm for interval graphs due to Booth and Lueker [13]. When given an interval graph G, this algorithm produces as a byproduct a *consecutive cliques arrangement (CCA)* of G - an ordered list of the maximal cliques of G such that each vertex belongs only to cliques that appear consecutively in the list. CCA is yet another characterization of interval graphs that is equivalent to an interval representation. This recognition algorithm runs in $O(n+m)$ time, but uses a relatively complex data structure called PQ-tree. There is an alternative $O(n+m)$ recognition algorithm that was later discovered by Corneil, Olariu, and Stewart [24]. This algorithm avoids using non-standard data structures and is simpler to implement, but has a much more complicated proof of correctness [104]. It is based on multiple sweeps of a modified LexBFS and instead of a CCA, it produces a (ch, cp^C) ordering when the input is an interval graph.

4.4.2 New graph class from a VO with a nice property

Recall the greedy coloring algorithm from earlier. This algorithm produces an optimal coloring of a chordal graph G when applied to a reversed perfect elimination ordering of the vertices of G. The same algorithm is also optimal when applied to a topological sort of a transitive orientation, i.e. a (cp) ordering, of a comparability graph [82]. In both cases, we have a special vertex ordering on which the greedy algorithm is optimal. It turns out that every graph has a vertex ordering on which the greedy coloring is optimal - the vertices being ordered based on their color in a particular optimal coloring, with vertices of the same color being ordered consecutively. Unfortunately, finding such an ordering, and more generally obtaining an optimal coloring, is NP-hard on general graphs. The cases of chordal and comparability graphs are different since we can construct the special vertex orderings efficiently.

Our ultimate goal is to be able to solve COLORING efficiently on as large a class of graphs as possible. In the context of the above discussion, one possible avenue is to look for a vertex ordering that generalizes the (ch) and (cp) orderings, but satisfies some additional properties which are restrictive enough so that not all graphs would have such an ordering. One such example is formed by the *perfect* orderings introduced by Chvátal in the early 1980s.

Definition 4.4.1. (Chvátal [21]) A vertex ordering $(v_1, v_2, \ldots, v_n)$ of graph G is *perfect* if, for each induced (ordered) subgraph F of G, the number of colors used by the greedy coloring algorithm on F is optimal on F. A graph G is *perfectly orderable* if it has a perfect vertex ordering.

Unfortunately, the recognition problem for the class of perfectly orderable graphs is known to be NP-complete [15]. A great deal of research has been done on exploring subclasses of perfectly orderable graphs for which there are efficient recognition algorithms, including *bipolarizable*, P_4-*indifference*, and P_4-*comparability* graphs. This was facilitated in part by the following characterization of perfect orderings.

Theorem 4.4.2. ***(Chvátal [21])*** *A vertex ordering $<$ of a graph G is perfect if and only if G contains no (induced) $P_4 = a - b - c - d$ such that $a < b$ and $d < c$.*

Note that the alternative characterization shown in Theorem 4.4.2 can be easily translated to the language of forbidden ordered subgraphs. Perfectly orderable graphs are exactly equal to

(p_{14}, p_{24}, p_{34}) graphs (see Figure 4.7). This, along with the fact that perfectly orderable graphs are perfect, has led Olariu to generalize Chvátal's original notion of perfect orderings by identifying all 40 sets of forbidden orderings of P_4 that define graph classes which are perfect [91].

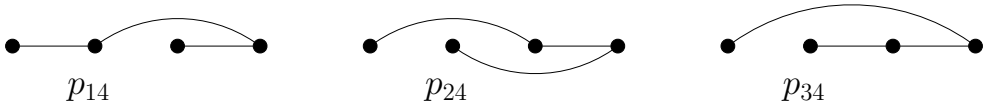


Figure 4.7: The forbidden ordered paths on four vertices characterizing perfectly orderable graphs.

We include another example of a simple algorithm for an optimization problem that gives rise to an interesting set of classes generalizing comparability graphs. Suppose G is a comparability graph and we are given a (cp) vertex ordering $<_G$ of G. To simplify the notation, assume the vertices in $<_G$ are labelled 1 to n. To find the size of a maximum clique in G, we can apply the following dynamic programming algorithm [104]. For each vertex i, denote by C_i the size of the maximum clique containing i as its last (rightmost) vertex in $<_G$. Let w be a neighbor of i preceding i in $<_G$. Then, since $(G, <_G)$ does not contain cp as an induced ordered subgraph, any neighbor of w preceding w in $<_G$ must also be a neighbor of i. This implies that to calculate the value of C_i, we simply need to find the maximum C_w over all neighbors w of i preceding i in $<_G$, and add 1.

An analogous dynamic programming algorithm can be used for CLIQUE on triangle-extendible graphs, if the (te) ordering is given [104]. In this case, instead of keeping track of the maximum cliques ending with a single vertex, we need to maintain the sizes C_{ij} of maximum cliques having i as second to last and j as last vertex in the (te) ordering. Then, we calculate C_{ij} by adding 1 to the maximum $C_{w,i}$, such that w precedes i and w neighbors both i and j. It is not difficult to see that, on an appropriate vertex ordering, we can further generalize the above dynamic programming approach to keeping track of the sizes of maximum cliques ending with an arbitrary constant number of vertices. This gives rise to the notion of *k-extendible* (or *k-clique-extendible*) graphs defined by having a vertex ordering that avoids *ke* - the ordered $k+1$-vertex graph with only one missing edge between the first and the last vertex. It follows that 2-extendible graphs are simply comparability graphs and 3-extendible graphs are the triangle-extendible graphs.

For k a constant, using the dynamic programming approach described above we can solve the CLIQUE problem on k-extendible graphs in $O(n^k)$ time when a (ke) ordering is given. An alternative dynamic programming algorithm that runs in $O(km^{k/2})$ time and may be more efficient on

sparse graphs, is presented in [58]. We know that we can recognize comparability graphs in time proportional to matrix multiplication. As mentioned earlier, Neogi et al. have recently shown that the recognition problem is NP-complete on both triangle-extendible graphs and on k-extendible graphs, for $k \geq 4$ [88]. Much like with perfectly orderable graphs, a possible next step is to consider subclasses of k-extendible graphs that could potentially be easy to recognize. We consider a couple of such candidate classes, that generalize permutation graphs, in Section 4.5.

Before proceeding we should include the following important remark. Establishing that recognition of a graph class is NP-complete does not immediately imply that a particular optimization problem on the class would necessarily be NP-hard. For example, intersection graphs of axis-aligned rectangles (*boxicity 2* graphs) and intersection graphs of paths in a tree (*EPT* graphs) are both NP-complete to recognize, but admit polynomial algorithms for the CLIQUE problem [104]. Therefore, the question of whether there exists a robust algorithm for the CLIQUE problem on k-extendible graphs – one that does not need to use a (ke) ordering – remains open.

4.4.3 Using a non-characterizing elimination VO to develop robust algorithms

We saw that the characterizing perfect elimination ordering of chordal graphs can be used to develop a simple polynomial algorithm for the CLIQUE problem. In particular, the VO allows us to reduce the CLIQUE problem on a chordal graph G to instances of the CLIQUE problem on a series of n induced subgraphs of G. Since all of these induced subgraphs are cliques themselves, each of the instances of the CLIQUE problem is trivially tractable. The solution of the initial CLIQUE problem on the overall graph then is simply the largest among the solutions to the instances of the CLIQUE problem.

A similar approach is also applicable to non-trivial settings where the special elimination ordering does not necessarily characterize the graph class. Consider the class of *unit disk graphs* consisting of intersection graphs of sets of circles in the plane having the same radius. Clark, Colbourn, and Johnson have shown that when the geometric intersection model (centers and radii of each circle) is given, the CLIQUE problem can be solved in polynomial time on unit disk graphs [22]. It was later shown that recognition of unit disk graphs is NP-hard [16]. Thus, one might think that the Clark, Colbourn, and Johnson model-based algorithm will not help in designing a robust algorithm for the CLIQUE problem on unit disk graphs. However, by focusing on certain steps of

the model-based algorithm which lead to efficient computation of the maximum clique, Raghavan and Spinrad were able to remove the dependence on the model [95].

Let G be a graph and let $e_1 < e_2 < \cdots < e_m$ be an ordering of its edges. We denote by G_i the graph formed by including edges which occur after edge e_i in the ordering. The edge ordering is called a *co-bipartite edge elimination ordering (EEO)* if for all edges $e_i = (x,y)$, the common neighbors of x and y in G_i induce a co-bipartite graph in G. Raghavan and Spinrad showed that each unit disk graph has a co-bipartite edge elimination ordering, thus reducing the original CLIQUE problem to m instances of the CLIQUE problem on co-bipartite subgraphs. CLIQUE is solvable in polynomial time on co-bipartite graphs. Moreover, it is easy to find a co-bipartite edge elimination ordering in polynomial time, if such an ordering exists – at each step greedily select and eliminate an edge whose endpoints' common neighbors induce a co-bipartite graph in the remaining graph (co-bipartite graph recognition is also polynomial). If we are not able to eliminate all edges with the above procedure, the graph does not have a co-bipartite edge elimination ordering and therefore cannot be a unit-disk graph. This constitutes a robust algorithm for CLIQUE on unit-disk graphs.

This leads us to our main result in this chapter regarding the CLIQUE problem on IP-SEG graphs. Recall Corollary 3.5.1 which stated that every IP-SEG graph G has at least one vertex v whose closed neighborhood $N[v]$ induces a permutation subgraph of G. Note the similarity to chordal graphs where each graph has at least one simplicial vertex. Like chordal graphs, and as any graph class with an intersection model, IP-SEG graphs are a hereditary class, meaning that $G - v$ remains an IP-SEG graph. This implies that an IP-SEG graph G must have an elimination vertex ordering $<$ such that for any vertex v, the subgraph induced by the neighbors of v that are to the right of it in $<$ must be a permutation graph. We call such an ordering a *permutation vertex elimination ordering (VEO)*.

While the permutation VEO is not a VOC of IP-SEG graphs, it does allow us to develop a robust polynomial algorithm for the CLIQUE problem. Find a vertex v whose open neighborhood is a permutation graph – we can afford to do this since recognition of permutation graphs can be done in linear time. If such a vertex does not exist, then the graph cannot be an IP-SEG graph. If such a v does exist, in linear time identify the largest clique containing v in the remaining graph by solving an instance of CLIQUE on a permutation graph, remove v and repeat. With this algorithm that runs in $O(n^2(n+m))$ time, we either find that the input graph cannot be an IP-SEG graph or

produce a list of cliques, the largest of which must be a maximum clique of the input graph.

Note the connection between the above robust algorithm and the model-based CLIQUE algorithm (see Algorithm 1) where we were also reducing the original problem to instances of CLIQUE on permutation subgraphs. In addition, the robust algorithm for CLIQUE on IP-SEG graphs can also serve as a robust algorithm for CLIQUE on circle graphs in which the open neighborhood of every vertex must be a permutation graph. It is also worth noting that class of graphs characterized by having a permutation VEO is *much* larger than the class of IP-SEG graphs. While the class of IP-SEG graphs has $2^{\theta(n\log n)}$ graphs on n vertices, each bipartite graph has a trivial permutation VEO and therefore the class of graphs with permutation VEOs contains $2^{\theta(n^2)}$ graphs on n vertices.

The robust algorithms for CLIQUE on unit-disk and IP-SEG graphs point to a more general approach. Let $\mathscr{G}$ be a class of graphs on which CLIQUE is solvable in polynomial time. Further, suppose we know how to recognize members of the $\mathscr{G}$ in polynomial time. Then, we can solve the CLIQUE problem in polynomial time on both 1) the class of graphs having a $\mathscr{G}$ vertex elimination ordering and 2) the class of graphs having a $\mathscr{G}$ edge elimination ordering. Algorithm 5 provides a sketch of the robust algorithm for the case of vertex elimination orderings.

Algorithm 5 Robust algorithm for CLIQUE on $\mathscr{G}$ VEO graphs

RobustClique(G)

```
1:  C_max = ∅
2:  while G is not empty do
3:     e = 0                                  # count the eliminated vertices in this iteration
4:     for v ∈ V(G) do
5:        if G[N(v)] ∈ 𝒢 then
6:           C: a clique of maximum size in G[N(v)]
7:           if |C| > |C_max| then
8:              C_max = C
9:           end if
10:          remove v and all edges incident with v from G
11:          e = e + 1
12:       end if
13:    end for
14:    if e = 0 and G is not empty then
15:       return input is not a 𝒢 VEO graph      # we cannot extend the 𝒢 VEO
16:    end if
17: end while
18: return C_max
```

4.5 Combining Properties

Recall the intersection of properties and vertex ordering characterizations of permutation graphs. Permutation graphs are the intersection of comparability and co-comparability graphs, but they are also the graphs having a vertex ordering that simultaneously avoids the ordered subgraphs cp and cp^C. In Damaschke's forbidden ordered subgraphs notation, we have the following equality:

$$permutation\ graphs = (cp) \cap (cp^C) = (cp, cp^C)$$

In both characterizations, we are essentially combining the same pair of properties. In the IPC method, the combination is performed by taking the set intersection of the larger classes of comparability and co-comparability graphs. In the VOC method, we are requiring that a single vertex ordering simultaneously satisfies the characterizing vertex ordering properties of comparability and co-comparability graphs.

Let $\mathscr{P}_1$, $\mathscr{P}_2$, be vertex ordering properties. Then we trivially have $(\mathscr{P}_1, \mathscr{P}_2) \subseteq (\mathscr{P}_1) \cap (\mathscr{P}_2)$. Damaschke [27] mentions this observation in the context of forbidden ordered subgraphs, but $\mathscr{P}_1$ or $\mathscr{P}_2$ may belong to any other type of vertex ordering properties, including, for example, pruning sequences of distance-hereditary graphs.

We know that

$$interval\ graphs = (ch, cp^C) = (ch) \cap (cp^C)\ and$$
$$split\ graphs = (ch, ch^{CR}) = (ch) \cap (ch^{CR}).$$

However, we also know that

$$threshold\ graphs = (ch, ch^C) \subsetneq (ch) \cap (ch^C) = split\ graphs.$$

Thus a natural question is: for which properties $\mathscr{P}_1$, $\mathscr{P}_2$, does the equality $(\mathscr{P}_1, \mathscr{P}_2) = (\mathscr{P}_1) \cap (\mathscr{P}_2)$ hold? Unfortunately, while we know of some simple sufficient conditions [27], this question remains largely unresolved even when limited to VOC's in terms of forbidden ordered subgraphs.

In what follows, we examine two pairs of properties whose VOC combination is a proper subclass of their IPC combination. We also look at how the different combinations might affect the

tractability of important problems.

4.5.1 Distance-hereditary and chordal: Proper subclasses of ptolemaic graphs and the bandwidth problem

Consider the properties of being distance-hereditary (DH), i.e. having a pruning sequence, and of being chordal (ch), i.e. having a perfect elimination scheme. Howorka [62] has shown that the class $(ch) \cap (DH)$ obtained by IPC combination coincides with that of *ptolemaic* graphs, originally defined using pairwise distances between vertices.

Definition 4.5.1. (Kay, Chartrand [72]) A connected graph G is **ptolemaic** if for any four vertices u,v,w,x of G, $d(u,v)d(w,x) \leq d(u,w)d(v,x) + d(u,x)d(v,w)$, where $d(a,b)$ denotes the number of edges in a shortest path between vertices a and b.

As was the case with combining the chordal and co-chordal properties, there are two different VOC combinations of the distance-hereditary and chordal properties: a pruning sequence that is also a perfect elimination scheme - (DH,ch); or, a pruning sequence that is also a reversed perfect elimination scheme - (DH,ch^R). Unlike the case of chordal and co-chordal, however, both of (DH,ch) and (DH,ch^R) are proper subclasses of the IPC class $(ch) \cap (DH)$.

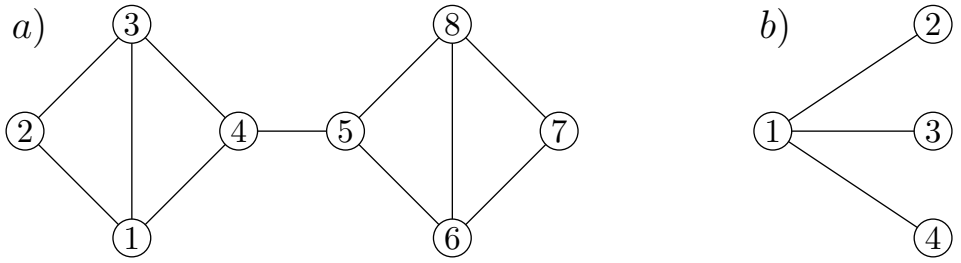


Figure 4.8: Ptolemaic graph that is not: a) (DH,ch); b) (DH,ch^R)

The graph shown in part $a)$ of Figure 4.8 is ptolemaic: (1,2,3,4,5,6,7,8) is a pruning sequence and (2,1,3,4,5,6,7,8) is a perfect elimination ordering. However, the only simplicial vertices of this graph are 2 and 7, none of which has a twin or is a pendant vertex. Thus this graph is not (DH,ch). Similarly, the *claw* graph shown in part b of Figure 4.8 has a pruning sequence (2,3,4,1) and a reversed perfect elimination ordering (1,2,3,4), so it is ptolemaic. But any pruning sequence must have 1 in position 3 or 4, while any reversed perfect elimination scheme must have 1 in positions 1 or 2. Therefore, the claw graph is not (DH,ch^R).

One of the relatively few optimization problems that are intractable on ptolemaic graphs is the minimum bandwidth problem. In fact, the problem remains intractable on trees, even on caterpillar trees of hair length at most 3 [30]. Note that the vertex ordering of any tree obtained by removing a leaf at a time is a (DH, ch) ordering. Thus, $trees \subseteq (DH, ch)$, which implies that the minimum bandwidth problem is intractable on the class (DH, ch).

The class of (DH, ch^R) graphs, however, is different. We not only have that the claw graph is not a member of this class, but also, no (DH, ch^R) graph could contain a claw as an induced subgraph. This means that the only trees that belong to (DH, ch^R) are simple paths. This opens up the possibility for the minimum bandwidth problem being tractable on (DH, ch^R) graphs. In fact, it is an easy exercise to show that any (DH, ch^R) vertex ordering cannot contain cp^C as an induced ordered subgraph. This implies that any (DH, ch^R) graph must be an interval graph. We have seen that the minimum bandwidth problem is tractable on interval graphs. The most efficient known algorithms include an $O(n\Delta^2 \log\Delta)$ algorithm by Muradyan (Δ denotes the maximum vertex degree), an $O(nBW(G))$ by Kleitman and Vohra, and an $O(n\log n)$ algorithm by Sprague [30]. Given that we do not know of an $O(n+m)$ minimum bandwidth algorithm for interval graphs, it would be interesting to see if such an algorithm exists for (DH, ch^R) graphs.

4.5.2 TE and co-TE: Two generalizations of permutation graphs

Recall that triangle-extendible graphs generalize comparability graphs and, by extension, co-triangle-extendible graphs generalize co-comparability graphs. We have also seen that $permutation\ graphs = (cp, cp^C) = (cp) \cap (cp^C)$.

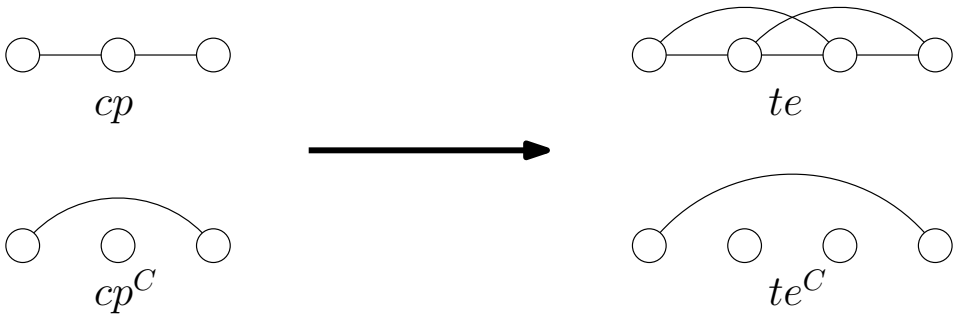


Figure 4.9: The forbidden ordered subgraphs characterizing (co-)comparability and (co-)triangle-extendible graphs.

Now consider the two graph classes (te, te^C) and $(te) \cap (te^C)$, obtained by applying the two

different methods, VOC and IPC, of combining the two properties TE and co-TE. Unlike the case with comparability and co-comparability, the VOC method leads to a class that is a proper subclass of the one obtained with the IPC methods.

Proposition 4.5.1. $(te, te^C) \subsetneq (te) \cap (te^C)$.

Proof. The 13-vertex graph G_{13}, illustrated in Figure 4.10, is a member of $(te) \cap (te^C)$, but not of (te, te^C). The vertex set of G_{13} is given by $V(G_{13}) = A \cup B \cup C$, where

- $A = \{a_1, a_2, a_3, a_4\}$ and A is an independent set,
- $B = \{b_{12}, b_{13}, b_{14}, b_{23}, b_{24}, b_{34}\}$ and B is a clique,
- $C = \{c_{12}, c_{13}, c_{14}\}$ and C is an independent set.

The additional edges of G_{13} that go between two different component sets are given by the following:

- b_{ij} is adjacent to a_i and a_j;
- c_{1i} is adjacent to a_1 and a_i;
- b_{1i} is adjacent to c_{1i}.

To simplify the proof, we will treat the two-digit indices of vertices in B and C as two-element sets, i.e. we wil treat b_{32} and c_{41} as alternative labels for the vertices b_{23} and c_{14}, respectively.

We consider the two vertex orderings $O_1 = (a_1, b_{23}, b_{24}, a_2, a_4, b_{34}, a_3, c_{12}, b_{12}, c_{13}, b_{13}, c_{14}, b_{14})$ and $O_2 = (a_2, a_1, b_{12}, c_{12}, b_{13}, b_{14}, b_{23}, b_{24}, b_{34}, c_{13}, a_4, c_{14}, a_3)$. A 4-vertex TE violation in a vertex ordering of G_{13} can only take one of the following forms:

- suborders with b_{1k} and c_{1k} appearing between a_1 and a_k, or
- suborders $a_i < b_{jk} < b_{lm} < b_{np}$ or $b_{np} < b_{jk} < b_{lm} < a_i$ such that $i \notin \{n, p\}$ and $\{j, k\} \cap \{l, m\} = \{i\}$.

The list of forms that 4-vertex co-TE violations in a vertex ordering of G_{13} can take is somewhat longer and includes suborders with:

- c_{1i} and c_{1j} appearing between b_{pq} and b_{rs}, where $\{\{1, i\}, \{1, j\}\} \cap \{\{p, q\}, \{r, s\}\} = \emptyset$,

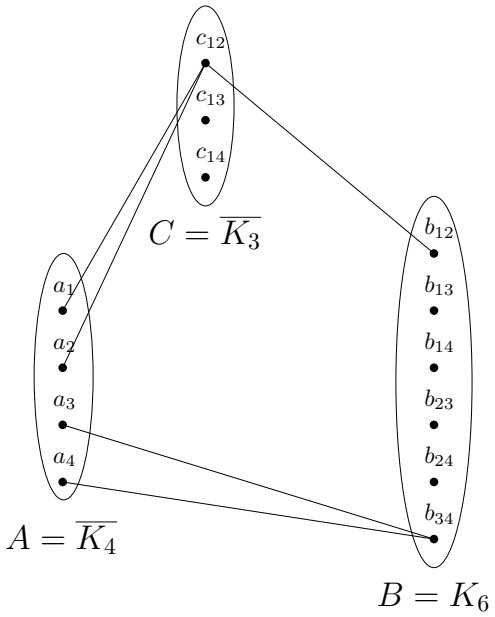


Figure 4.10: A sketch of the 13-vertex graph G_{13} that is $(te) \cap (te^C)$, but is not (te, te^C). The figure only illustrates a few of the edges going across component sets. The full set of edges is specified in the main text.

- c_{1i} and a_j appearing between b_{pq} and b_{rs}, where $i \notin \{1, j\}$, $\{1, i\} \notin \{\{p,q\},\{r,s\}\}$, and $j \notin \{p,q\} \cup \{r,s\}$,
- a_i and a_j appearing between c_{1k} and b_{1k}, where $\{i, j\} \cap \{1,k\} = \emptyset$,
- a_i and a_j appearing between c_{1k} and a_l, where $l \in \{1,k\}$ and $\{i, j\} \cap \{1,k\} = \emptyset$,
- a_i and c_{1j} (or c_{1i} and c_{1j}) appearing between c_{1k} and a_k, where $\{i, j\} \cap \{1,k\} = \emptyset$,
- a_i and a_j appearing between a_k and b_{lm}, where $k \in \{l,m\}$ and $\{i, j\} \cap \{l,m\} = \emptyset$, and
- c_{1i} and a_j (or c_{1i} and c_{1j}) appearing between a_p and b_{qr}, where $p \in \{q,r\}$, $p \neq 1$, $j \neq 1$, and $\{q,r\} \notin \{\{1,i\},\{1,j\}\}$.

Using the above enumeration of all forms TE and co-TE violations could take in a vertex ordering of G_{13}, it is easy to verify that O_1 is a valid (te) ordering and O_2 is a valid (te^C) ordering. This demonstrates that G_{13} does belong to the class $(te) \cap (te^C)$. However, O_1 contains the co-TE violation (a_1, a_2, a_3, c_{14}) and O_2 contains the TE violation $(a_1, b_{12}, b_{13}, b_{23})$. This means that neither O_1 nor O_2 is a valid (te, te^C) vertex ordering. In fact, using a computer program, we were able to verify that G_{13} cannot have a valid (te, te^C) vertex ordering, showing that $G_{13} \in ((te) \cap (te^C)) - (te, te^C)$.

□

The class of TE graphs contains all bipartite graphs and therefore it is a *superfactorial* hereditary class of graphs [103] – it contains $2^{\Theta(n^2)}$ graphs on n vertices. We show, however, that this is not the case for the classes of (te, te^C) and $(te) \cap (te^C)$ graphs [66]. Consider the following theorem.

Theorem 4.5.1. ***(Alekseev [4])*** *A hereditary class of graphs has $2^{\Theta(n^2)}$ members on n vertices if and only if it contains all bipartite graphs, all co-bipartite graphs, or all split graphs.*

Let B_{15} be the bipartite graph on 15 vertices with vertex partitions $X = \{x_i \mid 1 \leq i \leq 5\}$ and $Y = \{y_{i,j} \mid 1 \leq i < j \leq 5\}$ and edge set $E = \{(x_i, y_{jk}) \mid i = j \text{ or } i = k\}$. It is easy to see that this graph is highly symmetric – any permutation of $(1,2,3,4,5)$ implies a corresponding relabelling of the vertices of B_{15} that induces an automorphism of the graph. Therefore, if B_{15} has a co-TE vertex ordering, it would have to have a co-TE vertex ordering that has vertices x_1, x_2, x_3, x_4, and x_5 in that order. However, in whatever position we try to place vertex y_{15} in, relative to the vertices of the X partition, we would get at least one of (x_1, x_2, x_3, y_{15}) or (y_{15}, x_3, x_4, x_5) as a co-TE violation. Hence, B_{15} is a bipartite graph that is not co-TE and its complement $\overline{B_{15}}$ is a co-bipartite graph that cannot be TE. Note that the above argument did not depend on the fact that Y is an independent set in B_{15}. This implies that the split graph S_{15} obtained by taking B_{15} and making Y a clique also cannot be a co-TE graph.

$B_{15}, \overline{B_{15}}$, and S_{15} are examples of bipartite, co-bipartite, and split graphs, respectively, that do not belong to either of the classes (te, te^C) and $(te) \cap (te^C)$. If $f(n)$ is such the class of (te, te^C) (or $(te) \cap (te^C)$) graphs has $2^{(\Theta(f(n)))}$ members on n vertices, by Alekseev's theorem we must have $f(n) \in o(n^2)$. In addition, since both classes generalize permutation graphs, we also have $f(n) \in \Omega(n \log n)$. Finding tighter bounds on $f(n)$ is an open problem. We know that permutation graphs always have a "large" (of size $\theta(n^{1/2})$) clique or an independent set. It would be interesting to know if a similar result can be established for (te, te^C) or $(te) \cap (te^C)$ graphs. A positive result could help us in proving a tighter upper bound on the number of graphs in the respective graph class.

Most of the other natural questions are open on both of (te, te^C) and $(te) \cap (te^C)$ graphs. Given that (te, te^C) and $(te) \cap (te^C)$ generalize permutation graphs, an interesting question is whether the two classes have a nice geometric intersection or containment model related to their special vertex orderings. We would also like to know whether the two classes can be recognized in polynomial

time. In particular, a polynomial algorithm that constructs the relevant special orderings will imply a polynomial algorithm for the clique and independent set problems. Even if it were shown that recognition on (te, te^C) and $(te) \cap (te^C)$ is NP-complete, we would still have the related open question of whether there are robust algorithms for CLIQUE and INDEPENDENT SET on the two graph classes.

4.6 Conclusion

In this chapter, we looked at different types of vertex orderings and related graph classes. We explored how the properties defining some of the vertex orderings can affect the complexity of recognizing graphs in the class. Additionally, we looked at situations in which vertex orderings can be a powerful tool for designing efficient algorithms for optimization problems. In particular, we gave the first robust polynomial algorithm for the CLIQUE problem on IP-SEG graphs using the permutation vertex elimination ordering and discussed a generalization of the approach to vertex and edge elimination orderings where induced neighborhoods belong to classes of graphs on which the CLIQUE problem is tractable. Further, we considered two different methods – IPC and VOC – of combining two vertex ordering properties. We presented additional examples of pairs of properties for which VOC leads to a more restricted class of graphs than IPC and showed that in some cases this may lead to a class of graphs on which a generally hard optimization problem may become tractable. In what follows, we summarize several interesting avenues for future research, some of which were already touched upon in previous sections.

In Section 4.3, we mentioned that there are two main vertex ordering types: elimination orderings and orderings avoiding ordered subgraphs. We saw examples like chordal graphs which have VOC's of both types - perfect elimination ordering and (ch). We also saw examples like 3-transitive graphs or perfectly orderable graphs, which have a forbidden ordered subgraph characterization but, since the recognition problem on these classes is known to be NP-complete, they cannot have an elimination ordering defined by a local property that can be checked in polynomial time. It would be interesting to know which sets of forbidden ordered subgraphs lead to elimination orderings defined by local properties that can be checked in polynomial time.

An answer to the above question may be helpful in further understanding the complexity of recognition on graph classes with a forbidden ordered subgraph VOC's. In particular, it would

be interesting to extend the results by Duffus et al. [34]. In particular, since both the proof of Theorem 4.3.5 and the proof of Neogi et al. that recognition of TE graphs is NP-complete make use of the 3-COLORING problem, it would be interesting to see if the two proofs can be generalized to positively resolve Conjecture 4.3.1. Another approach would be to look at graphs *gn* that are not 2-connected and see if we can arrive at a more comprehensive set of sufficient conditions for recognition being polynomial on the class (gn), thus extending the work done by Hell et al. [61].

Another avenue for future research is to try to extend the work of Damaschke [27] and better understand when a multi-property VOC characterization is more restrictive than the corresponding IPC characterization, i.e. for which combinations of properties $(\mathscr{P}_1, \mathscr{P}_2, \ldots, \mathscr{P}_k) \subsetneq (\mathscr{P}_1) \cap (\mathscr{P}_2) \cap \cdots \cap (\mathscr{P}_k)$. Further, we would like to better understand how this is related to the complexity of recognition or different optimization problems on the corresponding classes. For example, we saw in Section 4.5 that bandwidth on $(DH) \cap (ch^R) = ptolemaic\ graphs$ is NP-complete, but we know how to solve it in polynomial time on the more restricted class (DH, ch^R).

Stemming from Olariu's work on perfectly orderable graphs [91], another possible direction would be to try to characterize, using vertex orderings, all graph classes on which a particular problem is (in)tractable. This is likely to be a challenging question and may need further refinement, but looking at a problem like bandwidth, that is about finding a special vertex ordering, may be a good starting point.

In this chapter, we saw examples of graph classes such as unit-disk and IP-SEG on which we can design a robust algorithm for the CLIQUE problem based on a special vertex or edge elimination ordering. We were able to do this because the corresponding neighborhood-induced subgraphs belonged to other graph classes on which recognition and CLIQUE are solvable in polynomial time. It could be worthwhile to more systematically examine classes on which CLIQUE and INDEPENDENT SET are still open and try to see if we can construct analogous elimination orderings.

Finally, it would also be interesting to study particular new graph classes defined using a VOC that generalize important existing classes. In particular, we would like to better understand the class of triangle-extendible graphs that generalizes comparability graphs. As suggested in Section 4.5, one possible approach is to first look at subclasses of this class obtained by intersection of properties – e.g. $(te) \cap (te^C)$ – or multi-property vertex orderings – e.g. (te, te^C).

CHAPTER V

Decentralized Consensus on Graphs with Different Structural Properties

5.1 Introduction

In earlier chapters, we discussed how properties of different models of restricted classes of graphs can help us design efficient algorithms for solving important optimizations problems such as COLORING, CLIQUE, and INDEPENDENT SET. Such algorithms are usually applied when we are interested in finding certain kinds of subgroups of entities or relations. In other settings, however, we may be interested in studying how different processes evolve or spread over a graph. In particular, we want to know if the dynamics of such processes change across restricted graph classes which may have different structural properties.

The processes of interest may include cascading failures in computer networks and power grids [25], disease spreads in contact networks of individuals [89], or information diffusion in social networks [75]. In some instances, the evolution of these processes may be the result of actions performed by, and interactions between, purposeful agents in control of vertices or edges of the graph. This usually takes the form of decentralized coordination between agents which may or may not have competing objectives [18, 53, 69, 85].

We note that in applied settings where graphs are used to model real, existing systems like the World Wide Web, individuals linked by family or friendship ties, or the chemical reactions taking place in a cell, the use of network science terminology is more common. Namely, these systems are considered to be networks of nodes with links between them, whereas their corresponding mathematical representations are graphs of vertices and edges [9]. We will interchangeably use the graph theory and network science terminology, without differentiating between graphs and networks.

In this chapter, we study the effects of the underlying network structure, across multiple graph classes and associated random generating models, on the effectiveness of different mechanisms for facilitating or disrupting coordination. We start by reviewing our work in [111] on the utility of communication on decentralized coordination between human subjects, including settings where competing incentives may be present. We also review the subsequent study of the disruptive effects of adversarial agents on decentralized coordination [56] and how well these can be mitigated

through the introduction of trusted agents. In both cases, we put emphasis on the varying levels of effectiveness of mechanisms for facilitating or disrupting coordination depending on the structural properties of the underlying graph model or class. We then build on the work done in [56], by performing simulations over a wider range of graph-generating models, including geometric-based models of special graph classes such as IP-SEG graphs, and explore in more detail the effects of graph structure on decentralized coordination success. We find that reaching decentralized consensus is challenging on graphs with geometric intersection models, including random geometric graphs and IP-SEG graphs. We show that the chances for consensus can be improved, over all considered graph classes and random generating models, by ensuring that the set of visible nodes is connected. Through simulations, we demonstrate that by increasing the trust that agents have in their visible neighbors, we can further increase consensus rates. Moreover, we find that there is an amplified positive effect when optimized placement of visible nodes is combined with increased trust in visible nodes. Finally, we discuss directions for future study, including alternative placements of visible nodes and additional graph classes and associated generating models.

5.2 Effects of communication and strategic tension on decentralized coordination

Coordination between individuals plays a central role in many biological processes, often to facilitate energy saving. Examples include huddling in groups of penguins, drafting in flocks of birds and schools of fish [108], and groups of ants working together to move a large object [84] or build a bridge with their own bodies allowing other members of the colony to cross a gap in the foraging trail [97]. Coordination is also ingrained in a wide range of human tasks, from navigation in large crowds and coordination of cockpit members [60], to coordination between disaster response teams on the ground and teams preparing to deploy [33, 100]. As a result, there is considerable literature that explores mechanisms that facilitate more successful coordination [33, 81, 98].

Communication is considered to be one of the key mechanisms promoting human coordination and has received significant attention in previous literature, including studies using human subjects [42, 105] and theoretical methods [29, 38]. In most of this literature, communication is only allowed in a distinct, pre-play stage, with particular focus on simple, two-player games. However, real tasks involve a significantly greater number of coordinating agents and communication takes place during the coordination phase. Further, in real coordination settings, the access to information and ability

to communicate are often localized.

In order to better account for such complexities, in [111], we investigate the effects of communication on a more complex coordination task involving 20 players. The players are situated as nodes of a network who can make decisions in real-time over a fixed time-horizon, but only observe decisions by their network neighbors. This setup builds on a class of games introduced by Kearns et al. [69, 73, 74], considering global consensus as the goal, but allowing agents to have clashing individual incentives. Unlike most prior literature, we allow subjects to communicate while they are making decisions in real time. Moreover, we explore the impact of a) communication restricted to immediate neighborhoods, b) global communication across the entire network, and c) communication with strong constraints on the information content of messages, on the ability of subjects to ultimately reach global network consensus, with or without strategic tension.

5.2.1 Experimental methodology

Echoing several prior experiments in networked consensus [69, 73, 74], we designed experiments in which sets of 20 human subjects were assigned to nodes of a network and were asked to choose one of two colors (red or green), with the primary objective of reaching a global consensus on a single color. Subjects could see the color choices made by their network neighbors (or white, if no choice had been made) and could change their own color at any point. The crucial novel element in these experiments was the integration of an instant-messaging-like communication interface (see Figure 5.1).

First we recruited workers from Amazon's Mechanical Turk, a crowdsourcing online platform. In the recruitment stage, each worker completed a simple three-question English language proficiency test, followed by a tutorial that explained each part of the experimental setup, the networked consensus task, and the web application interface. Once they completed the tutorial, participants were required to affirm (by checking a box in an online form) that they were over the age of 18 and that they had read and consented to the terms of participation. We also asked participants to indicate their time availability and if they would agree to be contacted for scheduled experiment sessions. Workers were able to participate in a full experiment session only after they have completed the recruitment stage.

In total, we ran more than 250 games (instances of the networked consensus task) over 6 experi-

ment sessions, involving 131 distinct participants. Each game could last at most 60 seconds, but was terminated as soon as consensus was reached. Each participant received a base payment of $0.15 per game. In addition, if the game reached consensus each player received a bonus. The magnitude of the bonus depended on two things: 1) whether the game involved individuals with color preferences, and 2) which color was chosen as consensus. In the first case, no matter which color was chosen in consensus, all subjects received $0.20. In the second case, if consensus was achieved in which all players chose the individual's preferred color, this participant received a $0.30 bonus; if, on the other hand, the less preferred color was chosen in consensus, the bonus to this individual was only $0.10. In all treatments involving color preferences (which constituted half of all treatments), exactly 10 nodes preferred each of the two colors. To ensure quality of the data, we treated games in which one or more participants did not choose a color at all as invalid, and removed these from consideration. In the end, we were left with 239 valid games which comprised our first analysis.

In these experiments, we systematically varied four factors: a) communication form, b) communication structure, c) color preferences, and d) network structure. Communication form involved three treatments: no communication, which provided our baseline, local communication, where individuals could only send messages to their immediate neighbors, and global communication, which allowed the individual's messages to be seen by the entire network. With communication structure, we controlled the extent to which exchanged messages were constrained through two treatments: unconstrained, in which arbitrary natural language (or otherwise) messages could be exchanged, and constrained, in which an individual could only send messages of the form "# of my neighbors choosing green, # of my neighbors choosing red". The "unconstrained" treatments did involve two character-limit constraints: we imposed a 10-character limit on each message, and a 50-character limit on all messages sent by a given node in a game. In the experiments, these character limits appeared to be quite generous. We considered settings with no color preferences, and those in which different participants faced conflicting preferences about colors (for example, some receiving a higher payout for a red, and others for a green, consensus); however, the number of subjects preferring each color was always equal.

Finally, we considered three categories of network structure. The first set of networks were generated using the Barabási–Albert preferential attachment model (BA) [8]. In this model, we start with a set of q_0 disconnected nodes and continue by successively adding one new node at a

time, connected to q ($q \leq q_0$) existing nodes with a probability proportional to the current degrees of the existing nodes. Formally, the probability p_i that the new node is connected to the existing node i is $p_i = k_i / \sum_j k_j$, where k_i is the degree of node i and the sum is taken over all existing nodes j. The BA model is an example of a scale-free model with a node degree distribution that follows a power law of the form $P(k) \sim k^{-3}$. In other words, BA networks generally contain large numbers of nodes with small degree, and a small numbers of "hubs", which are nodes with very large degree.

The other two sets of networks were generated using the $G(n,m)$ variant of the Erdős–Rényi random graph model (ER) [39], in which a graph is chosen uniformly at random from the collection of all graphs which have n nodes and m edges. In networks generated by the ER model, connectivity is entirely random and they have approximately binomial degree distribution with a central peak near the mean degree $\frac{2m}{n}$ and rapidly decreasing probabilities for degrees away from the mean.

In all our experiments, we used networks of size 20 and set $q_0 = q = 3$, which meant that the BA model always produced networks with 51 edges. To try to differentiate between the effects that network edge density – fraction of possible edges present in the network or $\frac{2|E(G)|}{n(n-1)}$ – and node degree distribution might have on the ability of subjects to reach consensus, we considered two sets of ER networks: i) ER-Dense (ERD) with $n = 20$, $m = 51$, and ii) ER-Sparse (ERS) with $n = 20$, $m = 26$.

5.2.2 Aggregate results: the value of communication and the importance of graph density

Prior research considering the role of communication in coordination has almost universally found that allowing people to communicate improves their performance. However, most such investigations were either not tightly controlled, were very small-scale, or allowed communication only during a distinct pre-play phase, in which all individuals were allowed to discuss the task. Our experimental setup aimed to more realistically capture the role of communication by embedding it within the task itself and varying its form (local vs global) and structure (unconstrained vs constrained). We found that local communication had no significant benefit over no communication: 60% of all games were solved (subjects reached global consensus) when no communication was allowed, while 61% of games were solved in the local communication treatments. On the other hand, allowing subjects to communicate beyond local neighborhood boundaries (global communication), led to an overall consensus rate of 83%, a significantly higher fraction than either no or

local communication ($p < 0.005$ for both comparisons). Moreover, Figure 5.2a) shows that global communication systematically dominates the other two forms (local and no communication) across network structures. This observation remains consistent in treatments with or without color preference incentives (where people obtain a higher payout for consensus on one color rather than the other). The difference is particularly significant in the sparser ERS networks, where global communication exhibits nearly double the success rate of local and no communication treatments.

Previous literature, as well as common intuition, suggest that natural language can significantly improve the chances of success in human coordination tasks. Therefore, a plausible hypothesis would be that imposing constraints on messages should significantly degrade ability of subjects to coordinate. However, we find that the opposite is true. Namely, approximately 67% of games with unconstrained communication were solved, compared with 77% of games solved when only a single type of message could be sent (counts of the two colors in one's neighborhood) (comparison was significant at $p < 0.05$). As Figure 5.2b) shows, this observation extends to both local and global communication settings. However, the primary difference between consensus rates arises in BA networks (see Figure 5.2c)).

Looking at Figure 5.2a) again, the consensus rates in games with dense ER networks appear to match those in games with BA networks, but players are less successful in games over sparse ER networks when communication is not global. While this is a very limited setting with only two levels of network density considered, the above observation agrees with previous literature in that higher density should improve the chances for successful coordination in networked settings. On the other hand, it appears that the presence of global communication can be quite helpful in overcoming the challenge posed by lower network density.

Previous research has also found that the existence of network hubs (high-degree nodes) may help facilitate coordination. While we do not observe this consistently across treatments - Figure 5.2a) shows virtually no difference between consensus rates over ER and BA networks of the same density - we do observe significantly higher consensus rates in games with constrained communication than in games with unconstrained communication, when the underlying network is generated using the BA model (Figure 5.2c)). In fact, this difference persists in local constrained communication settings (significant at $p < 0.05$). This suggests that the presence of network hubs can be helpful when combined with appropriate constraints on communication structure. In our

experimental treatments, in particular, hub nodes in BA networks were often able to observe the color choices of 65-85% of network nodes. Therefore, the constrained messages they were sending, describing the state of their neighborhood at time t, were very good approximations of the global network state at time t. In addition, their high degree may also explain why in games on BA networks, constrained communication offered significant improvement over unconstrained communication, even when nodes were only allowed to communicate locally - a message sent by a hub node to its neighbors will be received by a sizable majority of nodes.

5.2.3 Individual behavior

To better understand the aggregate findings discussed above, we developed a parametric behavior model, making use of the following parameters which we hypothesize were the primary observable drivers of individual behavior:

- **Game stage**: we divided the game into three stages, beginning, middle, and end; the latter two stages (middle, end) were represented as binary variables (the beginning becoming the default).
- **Number of neighbors (neighbors)**: the number of neighbors of a player.
- **Fraction of neighbors choosing a different color (opposite color)**: the fraction of a player's neighbors who are choosing a different color from the decision maker.
- **Relative excess of received messages promoting different color over the same color (opposite message)**: the count of messages received that suggest using a different color less the count of messages promoting the same color as currently chosen by the decision maker, measured over the previous 15 seconds.
- **Preference for currently chosen color (prefer current)**: whether the player would earn higher payout if their currently chosen color becomes the consensus choice.

We discretized time at 1 second intervals and used a logistic regression to predict the probability that an individual will change their color in the next 10-second interval. We developed 5 such models, one for no communication, and one for each of the four combinations of communication

forms (local vs global) and communication structure (unconstrained vs. constrained), with all variables normalized to facilitate cross-variable and cross-model comparison. The results are shown in Figure 5.3. We found that behavior is broadly consistent across the different settings. As one might expect, having a greater fraction of neighbors with and receiving more messages advertising the opposite color increases, while the player preferring their current color reduces the chances that the player will change their color, in all communication settings. An intriguing observation is that the prevalence of messages advertising the color not currently chosen have the greatest impact on an individual's decision to switch, in most cases far greater than any other factor. In fact, it appears to be the strongest factor in local communication, even though we have found it to offer little improvement in facilitating coordination. Similarly, the impact of such messages on decisions only seems to diminish as we introduce constraints. What this strongly suggests is that it is the information content of messages, rather than behavior in response to these, that explains our aggregate observations.

To explore this hypothesis, in [111] we developed a quantitative measure of *marginal information* about the global state conveyed by messages over time. With this measure we tried to capture how much closer to the global state (proportion of network nodes having the current majority color) a recipient's observed information is after receiving messages over a fixed unit of time than they were prior to these messages (based on both choices by immediate neighbors, as well as messages received in the past). We refer the reader to [111] for the formal definition of the measure. We include an informal description, through example, below.

We focused on messages that could be perceived as suggestions for recipients to stay with or change to a particular color. For a recipient in the unconstrained communication setting, such messages might include "choose red", "red wins", or simply "red" (all of them indicating that the recipient should stay with or change to red), but do not include messages like "Sam", "come on", or "ignore pay". For recipients in the constrained communication setting, we assumed that they would treat messages reporting more neighbors of one color ("3 red, 2 green") as suggestions to stay with or change to that color (red). Now, suppose that recipient Y's neighborhood was predominantly red or in the past she had mostly received messages suggesting red. However, more recently she had started receiving significantly more messages suggesting green and the actual global majority color is green. In this scenario, we consider the recent communication this player received to have high marginal information as it better indicates the current global state of the network than the player's

neighborhood or past received communication.

Using this this approach, we found that marginal information conveyed by messages over time is significantly greater in global communication than in local, particularly early in the games, which is partly due to the fact that more messages are received in global communication treatments. We also found that difference in marginal information is especially significant in sparse ER networks, explaining the rather dramatic advantage of global communication in such settings. We also observed that constrained communication games involved messages with higher marginal information earlier during the game for dense ER and BA networks, speeding up consensus. This partially explains why constrained communication settings achieved somewhat higher consensus rates.

5.2.4 Summary of findings

Much of prior literature has found that communication has substantial value in facilitating coordination. This seems quite natural when one considers the importance of communication in everyday small-scale coordination activities, ranging from who picks up the kids from school to how a particular complex task should be split among several workers. Game theoretic literature, in particular, has explored extensively the strategic role of "cheap-talk" communication, taking for granted the role it serves in providing valuable information about the state of the world. Our experiments explored communication as embedded in a networked coordination task, allowing subjects to make decisions and communicate in real time, and we systematically investigated the impact that different constraints on communication play in its value to the coordination task. From a behavioral standpoint, we found that people indeed "respond" to messages that they receive: specifically, they are significantly more likely to change their decision if it conflicts with received messages. This behavioral trait is consistent across all communication treatments. The key difference is how informative received communication is. When people can only discuss the task locally, little information about global state is ultimately conveyed. In contrast, global communication is clearly far more informative, and that ultimately leads to improved performance. We see some evidence that constraining the structure of local communication could add some value, but this appears to be limited to network settings containing well-connected hubs whose messages can relay a very good approximation of the current global state to a sizable portion of the population. The consideration of information conveyed in communication has not figured significantly in prior literature, even though realistic

communication contexts are, typically, local. Consequently, our findings suggest that unconstrained communication through local channels may be insufficiently effective in promoting global coordination, and entities, such as media and government, with the ability to reach a broad array of the population may have a critical role to play in facilitating coordination.

5.3 Decentralized coordination in the presence of adversarial and trusted agents

There has been considerable prior literature devoted to studying and modeling human behavior in networked coordination settings, including networked consensus [69, 73, 74, 111], coloring [69, 85], bargaining [18], and social dilemma games [53]. However, decentralized coordination often takes place in the presence of adversarial agents. For example, organizations attempting to coordinate on a strategy may also compete with other organizations, and coordination in combat mission planning and execution inherently faces adversarial entities in the form of enemy combatants. Additionally, in many settings adversaries attempt to exert their influence undetected, such as by bribing insiders, taking control of network nodes through cyber attacks, and spreading fake information [5]. Therefore, it is important to consider resilience to adversarial tampering with the decentralized coordination process. While existing research has devoted considerable attention to ensuring robustness to faults and attacks, it has mostly done so assuming that non-malicious agents follow simple stylized models of individual behavior [2, 14, 79]. However, many settings feature human actors who play an important role in reaching consensus.

In this section, we will discuss one of the first studies to address the question of human behavior in adversarial coordination settings by Hajaj et al. [56]. This study investigates the problem of decentralized consensus on networks in the presence of adversarial nodes, first using human subject experiments with 556 participants, and subsequently through the data-driven agent-based modeling (DDABM) methodology [113]. The experiments explore two design factors and their potential to mitigate adversarial influence: allowing neighboring nodes to communicate and embedding a small set of trusted nodes in the network. Note that in [111], discussed in Section 5.2, we found that communication among network neighbors has limited value in facilitating consensus. On the other hand, much prior research, using stylized models of individual behavior, has argued that the presence of trusted nodes can significantly facilitate decentralized coordination [1, 2, 109].

The results in [56] that we will discuss in this section, run counter to both of these observations.

First, they demonstrate that local communication can help a great deal, especially as the number of adversarial nodes is increased. Second, they show that the presence of trusted nodes does not, in the aggregate, help, reinforcing the need to develop better models of individual and collective behavior in such settings. Next, Hajaj et al. develop a data-driven agent-based model of adversarial decentralized consensus on networks, following the DDABM methodology [113]. In DDABM, individual agent models are derived from data, and are then instantiated in an agent-based framework via features that capture behavioral inter-dependencies among network neighbors. These agent models serve three purposes. First, like the descriptive models of individual decision-making in [111], they provide further insight into individual behavior. Second, the resulting agent-based model effectively captures the experimental observations at the macro level, and is quite robust to small errors in the individual agent models. Third, it demonstrates the usefulness of the derived computational platform as a means for further simulation-based investigation of the adversarial consensus problem. This last point will form the foundation of our work presented in Section 5.4, where we explore the influence of network structure on decentralized coordination over a much wider range of network generating models, including some restricted to special graph classes.

5.3.1 Experimental methodology and aggregate results

The general experimental setup of Hajaj et al. in [56] largely replicated that in [111] which we discussed in Section 5.2. There was, however, one change that turned out to be quite consequential. The visual game interface in [111] featured a progress bar, showing how close the overall state is to global consensus (measured by the percentage of nodes having the majority color). In the new setting, however, such a progress bar communicated too much information, particularly when adversaries were present, and it was therefore removed (particularly since it did not have a clear motivation and was just a design artifact of prior sets of experiments [69, 73]). Removing the progress bar increased the importance of local unconstrained communication (there were no treatments with global or constrained communication in these experiments), relative to our earlier findings in [111].

The above experimental framework was augmented with several features in order to study how adversarial nodes impact the ability of the rest (i.e., the non-adversarial sub-network) to reach global consensus. Namely, players were divided into two teams: a *consensus* team and a *no-consensus* team (in our parlance, these are the adversaries). The goal of the *consensus* team is to reach global

consensus among members of this team only. The goal of the *no-consensus* team is to prevent consensus among members of the consensus team. The bonus payout of each player was contingent on their team achieving its goal. Unlike in [111], there were no treatments with competing incentives among players of the same team. At the beginning of the game, each player was assigned to one of these teams, and this assignment was indicated in their view of the game.

The number of *consensus* players in each game was fixed to 20, to control the baseline difficulty of the task (the underlying consensus problem on networks becomes more difficult as the network size grows, other things being equal). In addition, in each game there were *a no-consensus players* introduced, where $a \in \{0, 2, 5\}$. The value of a was not disclosed to the players at the beginning of a game; although an omniscient observer can infer it from the size of the overall network (which is $20 + a$), no player could, in fact, do this, since players could only observe their direct neighbors, and the maximum node degree was limited to 15 to facilitate effective visualization. Adversaries (*no-consensus* nodes) were non-visible to others, including other adversaries.

Additionally, it is often possible to have a small number of known *reliable* or *trusted* nodes on the network, for example, nodes which are particularly difficult to compromise due to a high amount of investment in their security, and conventional wisdom is that such nodes can greatly facilitate consensus [2]. To allow for this, *visible* members (henceforth, *visible nodes*) of the *consensus* team were introduced. The number v of *visible nodes* was also varied, with $v \in \{0, 1, 2, 5\}$. These nodes were visible only to their immediate network neighbors, which in the game interface was highlighted by an orange circle around the corresponding nodes. As in [111], three network structures were explored: BA, ER-dense, and ER-sparse. Average degrees increased slightly due to the addition of adversarial nodes. For example, BA and dense ER networks on 20 nodes had a mean degree of 5.1, whereas those on 25 nodes had a mean degree of 5.28. Nevertheless, the overall goal of differentiating between the effects of network density and those of degree distribution on decentralized coordination, remained.

The human subjects - 556 in total - were again recruited through the Amazon Mechanical Turk platform and they jointly played 1,080 games. The measure of coordination success used was the consensus rate - the proportion of games reaching global consensus on a single color among members of the *consensus* team. We now give a brief overview of some of the aggregate experiment results.

Confirming intuition, having adversarial players participate in the game did have a statistically significant deleterious impact on consensus rate ($p < 0.01$). Delving deeper, they distinguished between two kinds of impact adversaries can have: *structural* (the presence of adversarial nodes may lead to the subgraph formed by *consensus* nodes being poorly connected or disconnected) and *behavioral* (color choices made or messages sent by adversaries may mislead neighboring *consensus* nodes). Their analysis confirms that in the experiments with human subjects, adversaries can indeed have both kinds of negative impact on coordination between *consensus* nodes.

Hajaj et al. also find that allowing unconstrained communication between neighbors has a clear positive impact on reaching consensus: when no adversaries are present, communication increases consensus rate by 23.5%, with 2 adversaries improvement rises to 35.1%, and with 5 adversaries games that feature communication are 54.5% more likely to reach consensus than those that do not. In particular, they find that in settings where adversaries have significant structural impact (the subgraph of *consensus* nodes is disconnected), communication appears to neutralize the behavioral impact of adversaries. They note that communication helps even when there are no adversaries, in contrast with prior results [111]. The key distinction, as mentioned earlier, is the absence of the progress bar: without this source of global information, communication becomes considerably more informative.

In terms of the impact of network structure, the main observation made by Hajaj et al. is that BA are more resilient in the presence of adversaries. In particular, when no communication is allowed, the addition of adversaries leads to drop in consensus rates across network structures, but it is less pronounced on BA networks than on ER-dense networks. Moreover, when local unconstrained communication is allowed, 2 adversaries are unable to significantly impact consensus rate, suggesting that when only few adversarial nodes are present, the ability to communicate endows scale-free networks with resilience even in the face of *behavioral* manipulation [56].

Finally, prior research that used stylized models of node behavior had demonstrated that the presence of trusted nodes can significantly improve the resilience to attacks [1, 2, 109]. However, in this experimental setting with coordination between human participants, Hajaj et al. find that the value of trusted nodes is limited. They explain that typical models assume that trusted nodes cannot be attacked, whereas in this setup, trusted nodes (as any other node) have no information about who the adversaries are and can also be influenced by the attackers. Nevertheless, they do

find that allowing players (including the trusted node) to communicate can improve resilience when there are many adversarial nodes.

5.3.2 Modeling of individual behavior and agent-based simulations

In [111], we followed the analysis of aggregate experiment results by developing models of individual behavior to better understand what are the main factors affecting when players decide to change their color (after they have made an initial choice). Hajaj et al. also proceed by developing models of individual behavior, but in addition to being interested in describing individual behavior, they specifically tailor the models to allow for agent-based simulations both with the goal of both validating the models on a macro level (how closely aggregate simulation outcomes match aggregate experimental results) and providing a computational platform for additional investigation of decentralized coordination on additional treatments not explored in experiments with human subjects.

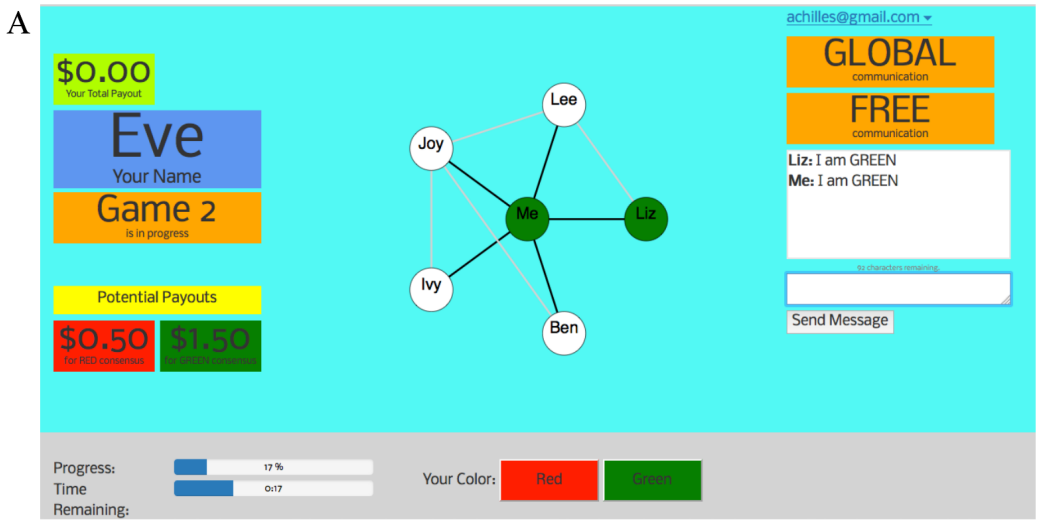
Given that communication in these experiments allowed for messages without constraints on their structure and it is not clear how to model this in a simulation, the authors focus on the setting with no communication. This, combined with the fact that players only choose between two colors, might make one think that the modeling task ahead should be straightforward. There are several complications, however. The first is that individual nodes have one of three roles: adversarial, visible, and regular (members of the *consensus* team that are not visible). It is intuitive that adversarial nodes would behave differently from others, but as the models below will show, there is also a difference in behavior between visible and regular nodes, even though they are both members of the *consensus* team. Second, nodes in any of the above three roles may behave differently depending on whether they see visible nodes in their neighborhood. Finally, the third challenge involves modeling real-time color choices by players. To address the first two challenges, 6 individual agent models were created: two models for each role, one of which was for situations with visible neighbors and another for situations without any visible neighbors. The third challenge was addressed by discretizing time into 1 second intervals, leading to at most 60 decision points in a game.

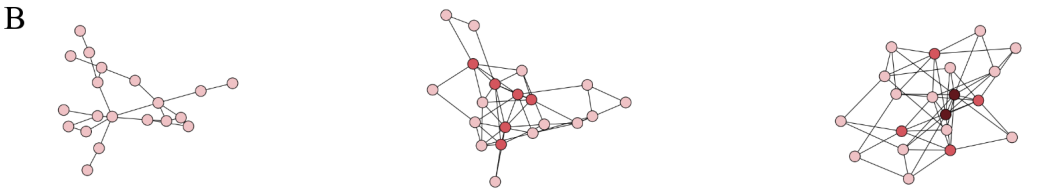
An additional complication that needed to be addressed was the fact that agents make two kinds of decisions during the span of a game: first, as they start as "white", they must choose an initial color, and subsequently, they choose whether to switch their color. The initial decision is a deliberate choice of a particular color, and includes both the timing of the choice, as well as the particular

choice between red or green color. Once they have chosen an initial color, players exhibit a considerable amount of inertia - they change color less frequently than once every 20 seconds on average. Given the inherent symmetry of their decision at this point (players do not have a preference for one color over the other beyond reaching consensus), it can be modeled as the timing of their next color switch. Consequently, the authors developed and trained 3 different models for each of the 6 combinations of roles and neighborhood assignments: 1) *timing of initial color choice*, 2) *initial color choice*, and 3) *timing of color switch*. We refer the reader to [56] for additional detail regarding the models. Below we give a brief description of the features being used.

- D_t: absolute difference between the fraction of player's neighbors of type t that picked red and the fraction that picked green.
- N_t: number of a player's neighbors of type t.
- G_t: fraction of a player's non-visible neighbors of type t choosing green.
- R_t: fraction of a player's non-visible neighbors of type t choosing red.
- O_t: fraction of a player's non-visible neighbors of type t choosing the opposite color from the one chosen by the player.
- C_t: fraction of a player's non-visible neighbors of type t choosing the same color as the player.

Note that features will have two instances, based on the type of neighbors they are referring to: *vis* for neighbors that are visibly on the *consensus* team and *inv* for non-visible neighbors whose role (regular or adversarial) is not discernible to the player. Additionally, O_t and C_t are only meaningful when the player has made an initial color choice and are only used in the *timing of color change* models. The subsets of features used by individual models can be seen in Tables 5.1, 5.2, and 5.3 (these are Table 1, 2 and 3 in [56], respectively). Note that the absence of a coefficient value indicates that the column feature is not used by the corresponding row's model. Each of the 18 models is represented by a logistic regression with these features, the coefficients of which were learned from experimental data. Also, l_1 (sparse) regularization was added to control for overfitting, with a regularization parameter tuned using cross-validation.



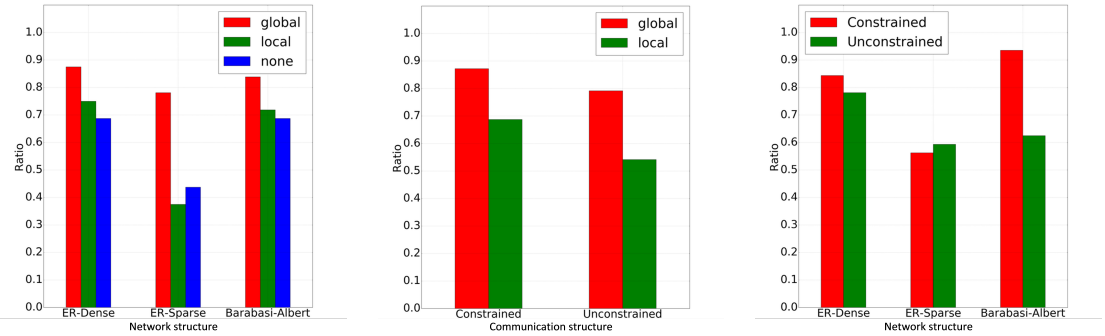


ER-Sparse ER-Dense BA

Figure 5.1: **A**: an example graphical interface from the point of view of an experimental subject, who is represented by a node in the network. The subject can see both her own node (labeled as "Me") and her network neighbors (labeled with their pseudonyms, randomly assigned at the beginning of a game), as well as connections among her neighbors. The subject can also observe her current total payout in the experiment session (over all games played thus far). In some treatments individuals had preferences about which color is chosen in consensus; these preferences in terms of the potential payouts are shown on the left. Also on the bottom left portion of the interface the subjects see both progress towards global consensus, as well as time remaining in the game. Finally, in games involving communication, an instant-message-like interface is shown on the right, with a box where messages can be viewed and entered. A clearly labeled sign describes whether the player is allowed to engage in LOCAL or GLOBAL communication. **B**: example instances of networks generated by the 3 models considered, where darker colors indicate higher node degrees. This figure appears in [111] and can be accessed at https://doi.org/10.1371/journal.pone.0170780.g001.

Table 5.1: Timing of initial color choice, P(choose a color).

Type	VN	Intercept	D_{inv}	D_{vis}	N_{inv}	N_{vis}
Reg	No	-1.952	1.290		0.000	
	Yes	-2.210	0.548	0.933	0.002	0.016
Vis	No	-2.045	1.742		0.040	
	Yes	-1.734	0.579	0.840	-0.061	0.048
Adv	No	-2.284	1.250		0.011	
	Yes	-2.744	0.802	0.662	0.025	0.155



a) b) c)

Figure 5.2: a) Differences in consensus rate under different communication forms grouped by network structure. b) differences in consensus rate for constrained and unconstrained communication, in local and global treatments. c) differences in consensus rate under different communication structures grouped by network structure.

	None	Local	Global	Local (constrained)	Global (constrained)
Intercept	−1.61***	−27.71***	−17.35***	−10.85***	−6.90***
Mid-game	0.05	0.27***	−0.09*	0.05	0.25***
End-game	−0.51***	−0.49***	−0.91***	−0.53***	−0.88***
Neighbors	−2.05***	−1.28***	0.26*	−0.63***	0.32*
Opposite Color	2.20***	2.43***	2.23***	2.54***	2.45***
Opposite Message	NA	43.30***	24.70***	14.83***	7.06***
Prefer Current Color	−0.33***	−0.34***	−0.02	−0.56***	−0.26***

Overall, coefficients are qualitatively consistent across treatments, suggesting that it is the information conveyed in messages that is largely responsible for our aggregate findings.

* $p < 0.1$

** $p < 0.01$

*** $p < 0.001$.

Figure 5.3: Coefficients of a logistic regression model of individual behavior, relating different aspects of the information available to a player to the likelihood of that player changing their current color. This figure appears in [111] and can be accessed at https://doi.org/10.1371/journal.pone.0170780.t002.

Table 5.2: Initial color choice, $P(\text{red} \mid \text{choose a color})$.

Type	VN	Intercept	G_{inv}	G_{vis}	R_{inv}	R_{vis}
Reg	No	0.000	-4.863		5.032	
	Yes	-0.066	-2.855	-2.022	3.453	1.733
Vis	No	0.109	-4.411		4.202	
	Yes	0.188	-3.215	-1.599	2.395	1.996
Adv	No	-0.023	0.817		-0.649	
	Yes	-0.286	0.172	0.732	-0.204	0.000

Table 5.3: Timing of color switch, $P(\text{switch color})$

Type	VN	Intercept	O_{inv}	O_{vis}	C_{inv}	C_{vis}	N_{inv}	N_{vis}
Reg	No	-3.98	2.65		-0.33		-0.01	
	Yes	-3.79	1.10	1.48	-0.87	0.09	0.00	-0.03
Vis	No	-4.11	2.70		-0.10		-0.01	
	Yes	-3.53	1.07	1.27	-0.33	-0.29	-0.06	0.00
Adv	No	-2.80	-1.13		1.19		0.00	
	Yes	-2.72	-0.60	-0.37	0.95	0.30	0.00	-0.20

Looking at the trained models, we can say that they broadly agree with expectations. Players on the *consensus* team tend to choose the color that a majority of their neighbors currently have, while adversarial players tend to go in the opposite direction, trying to disrupt the formation of a clear majority in their neighborhood. We include a couple of observations regarding the decision-making of individual players, made by Hajaj et al. that were somewhat unexpected [56]. First, for players on the *consensus* team, disagreement among visible neighbors has a more significant, positive impact on the likelihood of them choosing an initial color at a particular time point. Adversarial players, on the other hand, seem to react more strongly to disagreement between non-visible neighbors (see Table 5.1). These dynamics can also be observed in the *timing of color switch* models (Table 5.3), with *consensus* team members reacting more strongly to the decisions made by their visible neighbors and adversarial players being more influenced by the decisions of their non-visible neighbors. Second, when deciding whether to choose red or green as their initial color (see Table 5.2), adversarial nodes appear to act relatively unaggressively: the negative relationship between neighbor choices and their own initial color choice is relatively slight, in comparison with the magnitude of the positive relationships for the regular nodes (note that the values of all features used by the *initial color choice* models fall in the range $[0,1]$, so this comparison is meaningful). This unaggressiveness of adversaries continues to persist even after they have made their initial choice and they are deciding when to make a color switch (see Table 5.3). The authors conjecture that the explanation for this is that adversaries are attempting to achieve their disruptive goals without being overly obvious to their non-adversarial neighbors.

Hajaj et al. validated their models individually, based on how good they were in predicting the decisions actually taken by human subjects finding themselves in the corresponding setup during

the experiments. In particular, they found that all 18 models were highly effective: they either exhibited high accuracy (90-95%), or large likelihood improvement over a frequency-based baseline (50%-100% improvement). In addition, they validated the collection of models as a whole, in terms of aggregate outcomes of agent-based simulations. In particular, for each of the environments experienced by human subjects in experiments, they constructed an agent-based model (ABM) with agents being instantiated as nodes on a network and each agent being assigned one of the three roles. The network and the role assignments are generated exogenously, using the same methods and parameters used to generate the corresponding experimental environment. After running agent-based simulations across the full set of experimental environments the authors found a reasonable quantitative agreement, with the largest deviation between simulation outcomes and the experimental consensus rates being within 0.14. The qualitative agreement was even stronger. Simulation consensus rates were dropping as the number of adversaries was increased. In addition, simulation consensus rates were highest on BA networks, closely followed by ER-dense networks, while the lowest consensus rates were observed on ER-sparse networks. They add one final validation step, where they show that their set of models are robust to perturbations in the model parameters.

After extensive validation, Hajaj et al. proceed by performing additional simulations using the above models, to explore settings that were not covered in the experiments with human subjects. Namely, in the experiments, roles were randomly assigned to nodes. In their simulations, the authors consider scenarios where the visible nodes or the adversarial nodes could be placed optimally (from their team's perspective) on the network. The heuristic used as a proxy to optimality is the number of unique neighbors that players with the corresponding role have. For example, if the *no consensus* team had the opportunity to try and maximize their potential influence before the game starts, one intuitive way to do it would be to position themselves on a set of nodes A whose union of neighborhoods is as large as possible (the ideal case being when A is a dominating set of the entire graph). The simulation results lead the authors to several interesting observation. First, it appears that even with optimized placement of visible (trusted) nodes, their value remains limited in settings with no adversaries. On the other hand, when adversarial nodes are afforded optimized placement, they are highly effective. In fact, this is true even if the "best" nodes were given to the visible nodes first, and adversarial nodes only had the chance to "optimize" their placement on the remaining nodes of the network. Nevertheless, the authors do observe that in simulated games with

adversaries, having 1 or 2 visible nodes placed optimally does help (compared to not having visible nodes at all), particularly on BA networks.

5.3.3 Summary of findings

In [56], Hajaj et al. consider the problem of adversarial consensus on social networks, with both conducting experiments with human subjects and performing agent-based simulations. Their experiment results show that the ability to communicate locally, can significantly improve coordination success despite adversarial presence, but randomly placing trusted nodes within the network is of limited value. They use experimental data to construct and validate an agent-based model of adversarial consensus, opening the possibility of exploring settings not covered in experimental treatments through agent-based simulations. Finally, they make the first steps in this direction by exploring settings where the placements of visible or adversarial nodes are not both random, and show that the ability of visible nodes to counter adversarial influence may be increased if their placement is optimized.

5.4 Further exploration of the effects of graph structure through simulations

In this section, we extend the work done in [56] by using agent-based simulations to explore decentralized networked consensus across a wider variety of treatments [65]. We take another look at structural properties such as density and degree distribution that affect the ability of agents to coordinate and explore what happens in certain extreme network examples. We explore additional node placements and consider potential individual behavior changes (simulated through model coefficient modifications) and we show that they can significantly increase the value of trusted nodes and, as a consequence, improve the chances for consensus. Finally, we examine decentralized consensus on an expanded set of network topologies and show that consensus can be particularly hard to reach on networks arising from geometric models.

5.4.1 Structural properties and extreme graph examples

Recall that the experiments in [111] and [56] explored the effects of network structure on decentralized consensus rates across two dimensions: density and degree distribution. Two levels of network density were considered: one in the range $[0.11, 0.14)$ (for *sparse ER* networks) and another in the

range $[0.22, 0.27)$ (for *BA* and *dense ER* networks). Similarly, two types of degree distributions were considered: normal (*sparse* and *dense ER*) and power-law (*BA*). The experimental results, agreeing with prior literature, suggested that increase in network density facilitates better decentralized coordination. Similarly, the presence of hubs in networks with power-law distributions, leads to consensus rates that are higher than in networks with normal degree distributions. Our first set of simulations offers a more fine-grained look at the role that degree distribution plays.

Consider the baseline setting with no adversarial or visible nodes, on networks of size 20. In line with expectations, on the lower end of the density spectrum we observe consensus rates of 26.3% on trees generated with the BA and 13.1% on trees generated with the ER model. The most extreme trees in terms of degree distribution are paths (almost uniform distribution) and stars (one hub, with all other vertices pendant to it). Unsurprisingly, we find that reaching consensus on a path of 20 vertices is extremely difficult, while the opposite is true in the case of a star graph (see Table 5.4).

Table 5.4: Consensus rates on trees with different degree distributions.

Tree type	Path	ER	BA	Star
Consensus rate	8.2%	13.1%	26.3%	94.3%

In [57], the authors looked at how the consensus rate on ER and BA networks changes as we vary the density from 0.1 to 0.5 and noted that BA networks are more conducive to consensus at lower density levels, but the difference becomes almost negligible as the density increases. Nevertheless, for both network models, increasing the density does lead to higher consensus rates. For completeness, here we consider the full density spectrum. In addition, we introduce denser generalizations of the path and star graphs to see what effects adding more edges to such extreme examples can have. We define as a *k-wide-path* (kwPATH) the graph on n vertices formed by taking a path on n vertices and adding all edges between vertices i, j such that $j - i = k$. We call the graph formed by a clique of size k and $n - k$ vertices, each of which is adjacent to all vertices of the initial clique, a *k-star* (kSTAR). Note that 1-wide-path and 1-star are simply a path and a star on n vertices, respectively. We also consider a slightly different generalization of the star graph – a *k-wide-star* (kwSTAR) – where instead of having a central clique of size k, we have only one vertex of degree $n-1$ and $n-1$ vertices of degree $2k-1$. The 2-wide-path, 2-star, and 2-wide-star graphs on 6 vertices are shown in Figure 5.4.

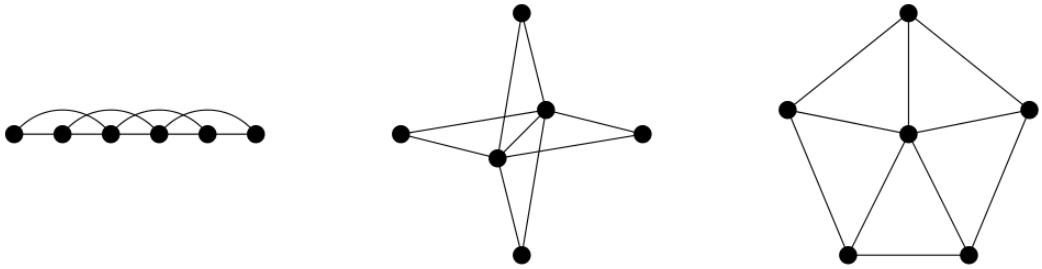


Figure 5.4: The 2-wide-path (left), 2-star (middle), and 2-wide-star (right) on 6 vertices.

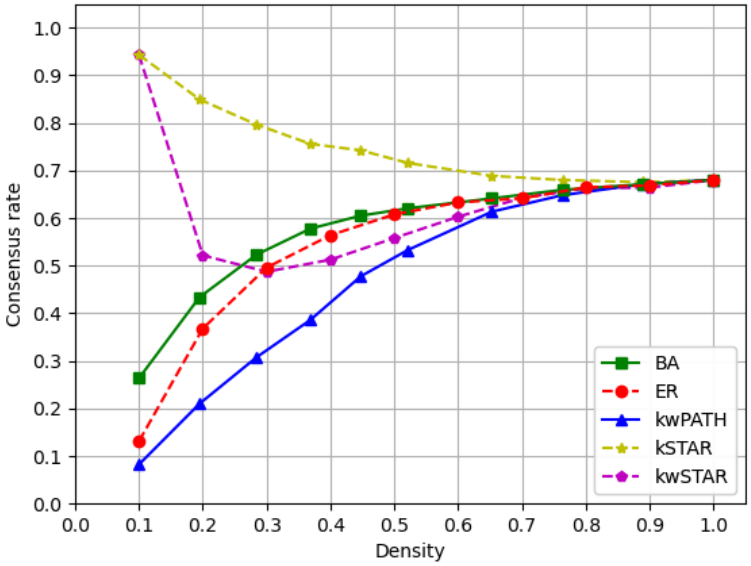


Figure 5.5: The effect that degree distribution has on the consensus rate, across different levels of graph edge density.

Figure 5.5 offers several interesting observations. First, adding edges to a graph, does not always lead to a higher consensus rate. In particular, when there are no adversaries present, having all nodes being connected only to a single hub is much better than having additional connections between the peripheral nodes (compare kSTAR with kwSTAR). Having more than one hub can also be somewhat detrimental (see kSTAR as we increase density). Finally, for very dense graphs, there is much less room for difference between the degree distributions and the consensus rate starts to level off.

5.4.2 Increasing the value of visible nodes

Prior work on decentralized coordination in stylized settings where the agent models are pre-defined (rather than trained on data from experiments with human subjects) has found that the presence of

visible nodes can significantly improve coordination [1, 2]. Experiment results in [56], however, found the presence of visible nodes being of limited value. Through the use of ABM simulations, the authors found that having visible nodes can somewhat improve consensus rates if their placement is optimized so that the union of their neighbors, or their *reach*, is as large as possible (i.e. the set of visible nodes is as close to a dominating set as possible). The improvement was particularly noticeable in BA networks, but nonetheless, was smaller than expected. The authors also noted that having too many visible nodes can be detrimental due to the increased potential for miscoordination among visible nodes themselves.

In this subsection we explore how the value of visible nodes can be further increased across two fronts. First, we show that when trying to optimize the placement of visible nodes, sacrificing some of the reach that visible nodes have (in terms of neighbors) to improve the connectivity between them, can lead to higher consensus rates. Second, through modifications of the individual behavior models, we find that human subjects may not be making the most out of the presence of visible nodes. We present several model modification approaches and argue that it is possible to further improve consensus rates by increasing the trust that players have in visible nodes through mechanisms such as additional pre-play training or modified coordination incentives. Further, we show that combining an optimized placement of visible nodes with adjustments to individual decision-making can have a compounding effect on coordination.

5.4.2.1 Placement of visible nodes: Connected dominating sets

In [57], the authors mention that the drop-off in consensus rates when increasing the number of visible nodes is likely due to more often having subgroups of visible nodes being disconnected from each other. In addition, an earlier work on a consensus protocol in the presence of visible nodes [2] identifies connected dominating sets as a particularly robust option for where to place visible nodes. Of course, this is not directly transferable to our decentralized consensus setting with human subjects as the agents, since we have a limit of 5 visible nodes, but networks will not always have a connected dominating set of size ≤ 5. Instead, we modify the notion of optimizing the placement of visible nodes to maximizing the reach that visible nodes have, under the constraint that the subgraph they induce is connected. To better isolate the effect that connectivity between visible nodes has on coordination, we also consider the setting where visible agents are positioned on the nodes of a

randomly selected connected subgraph. For the next set of simulations, we keep the rest of the setup as in the experiments in [56]: we consider BA, dense ER and sparse ER networks, and we consider settings with 0, 2, and 5 adversaries. Adversaries, if any, are still assigned randomly, after visible nodes have been placed. The results, aggregated by the number of visible nodes, are shown in Table 5.5.

Placement of visible	2 visible	5 visible
random	26.60%	27.21%
random | *connected*	30.71%	30.31%
max reach	35.12%	29.51%
max reach | *connected*	37.44%	32.62%

Table 5.5: Consensus rates on treatments with 2 and 5 visible nodes, under different visible node placements. In all settings, adversarial nodes are placed randomly.

First, we observe that ensuring connectivity, while trying to maximize the reach of visible nodes, is a slightly better approach than optimizing reach only. Second, ensuring connectivity between visible nodes helps even when we are not trying to optimize their reach. This may be useful in settings where increasing the reliability of multiple hub nodes may be too expensive, but we can still gain some coordination improvement by securing a set of lower-degree connected nodes. Finally, even though ensuring connectivity while maximizing reach does help, it does not resolve the drop-off in consensus rate we observe when going from 2 to 5 visible nodes. This suggests that the difficulty larger numbers of visible nodes have coordinating among themselves is only in part due to their placement and connectivity. How agents, including visible ones, behave in the presence of visible neighbors, may be another significant reason.

5.4.2.2 Model modifications: Increase trust in visible nodes

The coefficients of individual behavior models shown in Tables 5.1, 5.2, and 5.3, suggest that cooperative players try to strike a balance between what their visible and invisible neighbors have chosen when making their own decision. Given that a cooperative player's neighborhood may contain multiple adversarial players as invisible nodes, it is worthwhile considering how much weight a player should give to each type of neighbor when considering their own decision. Is it better for cooperative players to follow the overall majority in their neighborhood, especially since very often they would only have 1 or 2 visible neighbors? Could coordination improve if each cooperative

player P tried to more closely follow her visible neighbors and trust their judgement (which may be based on additional information from nodes that P is not able to see)?

We explore these questions through simulations using modified models of individual behavior. Since players spend most of their time in the game having already selected an initial color, we focus only on the color-changing models of cooperative players. Further, since we are interested in modifying the importance that cooperative nodes place on the choices of their visible neighbors, we focus only on the models for cooperative players that have visible nodes in their neighborhood. We consider model changes of the following kind. Given a tuning parameter t in the range $[0,1]$, we change the coefficients $(O_{\text{inv}}, O_{\text{vis}}, C_{\text{inv}}, C_{\text{vis}})$ to $((1-t)O_{\text{inv}}, O_{\text{vis}} + tO_{\text{inv}}, (1-t)C_{\text{inv}}, C_{\text{vis}} + tC_{\text{inv}})$. Intuitively, the larger the value of t is, the more importance a node gives to what their visible neighbors, as opposed to their invisible neighbors, do.

Note that increasing or decreasing an individual model coefficient may effectively lead to an increase or decrease in the frequency of color changes made by an agent. For example, only increasing the value of the O_{vis} coefficient, will effectively make the agents play the game at a faster pace, which will by default increase their chances of consensus (it is somewhat similar to giving players more time to complete the task). With the above model transformation in which we increase one coefficient, at the expense of another, however, we control for this - if a player P's neighborhood is fully red, for example, they would be equally likely to change their color for any value of t. This is in line with how the original models of players without visible neighbors relate to those of players with visible neighbors – there a portion of the value of O_{inv} is essentially transferred to O_{vis} when visible nodes are introduced.

Model variation	1 visible	2 visible	5 visible
$\boldsymbol{t = 0}$ **(no modification)**	26.56%	26.60%	27.21%
$\boldsymbol{t = 0.25}$	26.28%	27.91%	29.68%
$\boldsymbol{t = 0.5}$	27.32%	29.04%	31.38%
$\boldsymbol{t = 1}$	28.85%	30.60%	34.86%

Table 5.6: Consensus rates under different modifications to the color-changing models of cooperative players with visible neighbors, grouped by the numbers of visible nodes. Both visible and adversarial nodes are selected randomly.

As before, the remaining treatment aspects in these simulations followed the experimental setup from [56]. We can see from Table 5.6 that as we vary t from 0 to 1, there is some increase in the

consensus rate, it being more pronounced when more visible nodes are present, but this increase is not as large as one might expect. In particular, the increase when 2 visible nodes are present, even for $t = 1$, appears to be somewhat smaller than the increase we saw in Table 5.5 when we optimized the placement of visible nodes with respect to reach and connectivity. On the other hand, in games with 5 visible nodes, the t-based modifications to the model led to matching or slightly better consensus rates than placing the visible nodes optimally. In fact, we no longer see a drop-off in the consensus rate in games with 5 visible nodes, in contrast to what we observed in the earlier set of simulations which used the original models of individual behavior. This offers support for our hypothesis that miscoordination between visible nodes is also due to agents playing sub-optimally in the presence of visible neighbors. In other words, trusting visible neighbors even more is a better strategy than equally valuing the choices of visible and invisible neighbors.

5.4.2.3 Optimizing visible node placement and modifying agent behavior in the presence of visible nodes

The fact that optimized placement of visible nodes and modifications to agent behavior are both able to improve coordination, but achieve that by increasing the value of visible nodes across separate dimensions, leads to the natural question of what gain can we expect if we combine the two. Maintaining the same overall setup, we explore this question through simulations. Table 5.7 contains the results.

Overall, the results show a synergy between optimizing the placement of visible nodes and modifying the color-changing models of cooperative agents so that they value choices by visible neighbors more. These compounded improvements in consensus rates are noticeable even for $t = 0.25$. Notice that for visible nodes placed on a random connected subgraph, we no longer see a drop-off, but a sizeable increase, in the consensus rate when going from 2 to 5 visible nodes. In addition, when $t = 1$, we see that the highest consensus rate is achieved when both the number and reach of visible nodes are largest, as long as visible nodes are connected.

There are several factors at play here. The model variations facilitate better coordination between visible nodes, when visible nodes are connected. This helps the overall team of cooperative players to reach consensus as regular players are now trying to more closely follow the lead of a well-coordinated minority. At the same time, by ensuring connectivity between visible nodes, we

Model variation: $t = 0$ (no modification)			
Placement of visible	1 vis	2 vis	5 vis
random	26.56%	26.60%	27.21%
random | *connected*		30.71%	30.31%
max reach	32.72%	35.12%	29.51%
max reach | *connected*		37.44%	32.62%
Model variation: $t = 0.25$			
Placement of visible	1 vis	2 vis	5 vis
random	26.28%	27.91%	29.68%
random | *connected*		34.45%	36.13%
max reach	36.82%	39.64%	35.24%
max reach | *connected*		45.47%	42.44%
Model variation: $t = 0.5$			
Placement of visible	1 vis	2 vis	5 vis
random	27.32%	29.04%	31.38%
random | *connected*		37.68%	42.40%
max reach	40.32%	44.47%	40.85%
max reach | *connected*		53.52%	53.02%
Model variation: $t = 1$			
Placement of visible	1 vis	2 vis	5 vis
random	28.85%	30.60%	34.86%
random | *connected*		42.37%	53.44%
max reach	46.08%	49.60%	48.08%
max reach | *connected*		61.88%	68.76%

Table 5.7: Consensus rates across different combinations of visible node placements and model variations.

end up having more visible nodes that have other visible nodes as neighbors. By increasing the reach of visible nodes, we increase the number of regular nodes having other visible nodes as neighbors. If we think of the model variations as better strategies for making use of visible nodes, then the optimized placements (in terms of connectivity and reach) of visible nodes allow for more agents to take advantage of these strategies. The main takeaway from these simulations results is that the value visible nodes provide in reaching decentralized consensus can be significantly amplified, but it may require a combination of optimized placement and modifications to agent behavior.

While optimized placement of visible nodes can be easily transferred back to the experimental setting with human subjects as agents, doing the same for model modifications would not be straightforward. Instead of adjusting model coefficients, we would need to influence the behavior of cooperative human players so that they are more reactive to, and more trustful of, changes made

by their visible neighbors. The most obvious candidate mechanism at our disposal for influencing the behavior of cooperative players in this way is additional pre-play instruction and training. This could take multiple forms such as expanding the textual description of the experimental setup with a "suggested" strategy for playing the game (*follow your visible neighbors*) or including simulation results which show that being more responsive to and following the lead of visible players improves the chances for overall consensus. Another candidate mechanism is to adjust the coordination incentives for visible players – instead of having a single reward awarded when global consensus is reached, introduce a second, partial reward for when a consensus is reached between the subset of visible players only. While we can reasonably expect some change in subjects' behavior resulting from such additional steps, we cannot expect the change to be drastic. For example, values like $t = 1$ or $t = 0.5$ in the parameterized model modifications above would be difficult to attain. Additionally, it is possible that the added instruction and training may not lead to equal behavior changes among all subjects.

To address such concerns, we ran an additional set of simulations applying simpler (fewer coefficient adjustments), more limited (smaller coefficient adjustments) and variable (across different agents in the same game) modifications. Specifically, we consider the following four modifications to color-changing models of cooperative (regular and visible) players in the presence of visible neighbors:

- M_1: Change O_{vis} to $O_{\text{vis}} + 0.5$
- M_2: Change $(O_{\text{inv}}, O_{\text{vis}})$ to $(O_{\text{inv}} - 0.25, O_{\text{vis}} + 0.25)$
- M_3: For each cooperative agent A, select a value a uniformly at random from $(0, 0.5)$ and modify A's model by changing O_{vis} to $O_{\text{vis}} + a$
- M_4: For each cooperative agent A, select a value a uniformly at random from $(0, 0.25)$ and modify A's model by changing $(O_{\text{inv}}, O_{\text{vis}})$ to $(O_{\text{inv}} - a, O_{\text{vis}} + a)$

Note that all four models are more conservative in the number of feature coefficients they adjust. M_2 and M_4 affect two coefficients, while M_1 and M_3 affect only one coefficient. M_2 and M_4, on the other hand, perform the modification so as to reduce any impact on the average number of changes that a player makes in a unit of time. Finally, M_3 and M_4 perform modifications with slightly

different intensity for different agents, trying to account for the fact that efforts to influence behavior of human subjects may have non-uniform effects. A summary of the simulations results can be seen in Table 5.8.

Model variation: none						
Placement of visible	0 visible	1 visible	2 visible	3 visible	4 visible	5 visible
random	23.80%	25.87%	26.68%	27.16%	27.26%	27.90%
random | *connected*			30.54%	32.53%	30.94%	30.51%
max reach		33.06%	35.21%	34.49%	30.57%	28.84%
max reach | *connected*			38.71%	38.54%	36.08%	31.72%
Model variation: M_1						
Placement of visible	0 visible	1 visible	2 visible	3 visible	4 visible	5 visible
random		28.89%	32.13%	34.12%	36.71%	39.63%
random | *connected*			39.95%	45.16%	46.14%	46.43%
max reach		41.65%	47.83%	51.57%	47.75%	45.43%
max reach | *connected*			53.60%	58.98%	58.64%	54.88%
Model variation: M_2						
Placement of visible	0 visible	1 visible	2 visible	3 visible	4 visible	5 visible
random		25.80%	27.08%	27.92%	28.28%	28.68%
random | *connected*			33.18%	35.82%	33.75%	33.31%
max reach		35.01%	38.53%	38.39%	34.64%	32.11%
max reach | *connected*			42.57%	43.94%	41.42%	37.50%
Model variation: M_3						
Placement of visible	0 visible	1 visible	2 visible	3 visible	4 visible	5 visible
random		27.13%	28.92%	30.64%	32.32%	34.20%
random | *connected*			35.48%	38.67%	38.90%	37.67%
max reach		37.55%	42.07%	43.32%	39.53%	37.55%
max reach | *connected*			46.53%	49.05%	46.76%	43.01%
Model variation: M_4						
Placement of visible	0 visible	1 visible	2 visible	3 visible	4 visible	5 visible
random		25.84%	27.18%	27.53%	28.30%	28.59%
random | *connected*			31.27%	33.65%	32.62%	32.02%
max reach		34.10%	36.88%	36.08%	32.93%	30.10%
max reach | *connected*			40.68%	41.27%	38.67%	34.24%

Table 5.8: Consensus rates across different combinations of visible node placements and limited model variations.

Note that each of the considered model variations leads to higher consensus rates than the original models trained on experiment data, especially when combined with placing the nodes on a connected subgraph maximizing their reach. While smaller, these improvements are there even for M_3 and M_4 where different agents receive different levels of model adjustment. The overall takeaway is that consensus rates in adversarial decentralized coordination between human subjects in

networked settings can be improved through the use of trusted nodes. Further, these improvements can be amplified by optimizing the number of visible nodes and their placement and by increasing the importance that cooperative players place on trusted neighbors.

5.4.3 Decentralized consensus on graphs with geometric intersection models

Other settings where decentralized coordination between agents is an important problem are ad hoc networks and robotic swarm systems [43, 87]. Typically, wireless devices and robots in such settings have limited communication range and in simulations this is captured by placing the agents on nodes of a *random geometric graph*. The model of a random geometric graph consists of unit balls in the n-dimensional Euclidean space with randomly generated ball centers [43, 49], with two vertices u and v being adjacent if and only if the corresponding unit balls b_u and b_v contain the other's center. It is easy to see that this model is equivalent to an intersection model of n-dimensional balls with the same centers, but half the original radii. Thus, when $n = 1$, we simply have random unit interval graphs. Similarly, when $n = 2$, we have random unit disk graphs.

In addition for being good models of ad hoc networks and robotic swarm systems, random geometric graphs are also important in the context of social networks. Namely, in the study of network science it is known that many real-world social networks exhibit a *community structure* – the presence of dense subgraphs that are only loosely interconnected [92]. Further, real-world social networks also have higher *degree assortativity* – the tendency of nodes to be connected with other nodes of similar node degrees. While the Barabási–Albert preferential attachment model is able to capture some observed properties of real-world networks such as having a power-law degree distribution (i.e. being scale-free), BA networks do not exhibit a community structure or degree assortativity. Erdős–Rényi random graphs also have these shortcomings. Random geometric graphs, on the other hand, are not scale-free, but do have clear community structure [92] and high degree assortativity [6].

In this subsection, we explore the difficulty of reaching decentralized consensus on several classes of graphs with geometric intersection models. We consider random geometric graphs for $n \in \{1, 2, 3\}$, as well as four other classes that we have seen in earlier chapters – interval, permutation, simple IP-SEG, and IP-SEG graphs. As a baseline, both for the consensus rate and the graph density, we use dense ER networks from the original experimental setup.

Since we run simulations on only a limited number of graphs from each class, we also need to specify a corresponding random generating model, much like the $G(n,m)$ Erdős–Rényi random graph model that we use for dense ER graphs. One approach would be to leverage the $G(n,m)$ model and only keep a graph if it belongs to the graph class of interest. However, since we do not yet have a full polynomial recognition algorithm for IP-SEG graphs and recognition of unit-disk graphs is known to be NP-hard, we instead use random generating models based on the geometric intersection models of the graph classes.

In particular, for random geometric graphs we choose the centers of the n-dimensional balls uniformly at random from within the n-dimensional unit cube $[0,1]^n$ [43]. Since we want to control for density, we scale the radius of the n-dimensional balls down to $0.07, 0.165$, and 0.24 for n being equal to $1, 2$, and 3, respectively. One random generating model for interval graph introduced by Scheinerman [101] generates n interval segments by randomly selecting the $2n$ segment endpoints from the interval $[0,1]$. However, the expected density of these graphs is $\frac{2}{3}$, making this approach less suitable for generating graphs with edge density in the range $[0.22, 0.27)$. Instead, we use a different model, also due to Scheinerman [102], in which we first randomly select n points in the interval $[0,1]$ to serve as centers of the n interval segments. Then, for each segment, we randomly select a radius r in the range $(0,\rho]$, where ρ is the parameter of the model that allows us to better control for density. In these simulations, we use $\rho = 13$.

We are not aware of any random generating model for permutation graphs in prior literature that allows controlling for expected edge density. Therefore, we simply use Scheinerman's parametric model for interval graphs to identify the x coordinates of segment endpoints. Then, for each segment, we randomly choose which of the endpoints will be placed on $\mathscr{L}_1$ and which on $\mathscr{L}_2$. Since simple IP-SEG and IP-SEG models consist of interval and permutation segments, we again take advantage of Scheinerman's parametric model, with an additional initial step that randomizes the numbers of permutation and interval segments in the model. To control for the expected density in permutation, simple IP-SEG, and IP-SEG graphs, we use slightly different values for the parameter ρ – 0.21, 0.2, and 0.26, respectively.

Note that, all of the above models, except the $G(n,m)$ Erdős–Rényi random graph model, control for expected edge density. In addition, none of the above random generating models is guaranteed to always produce a connected graph (in contrast to the BA model, mentioned earlier). Therefore,

in preparing our simulation settings, we repeatedly generate graphs from each model until they are both connected and have exactly the same number of edges as the dense ER graphs from the experimental setup, namely $3(n-3)$, where n is the number of vertices. As in the experimental setup, we consider settings with $0, 2$, and 5 adversaries. For conciseness, we only look at the baseline (random placement and no model modifications) and the best placement and model modification combination from the previous set of simulations (*max reach* | *connected* and M_1).

No model modification - random placement						
	0 visible	1 visible	2 visible	3 visible	4 visible	5 visible
ERD	26.28%	29.40%	32.06%	32.48%	32.21%	33.12%
GEO, $n = 1$	16.20%	17.32%	18.37%	18.70%	20.01%	19.97%
GEO, $n = 2$	18.28%	20.19%	21.78%	21.87%	21.60%	22.17%
GEO, $n = 3$	20.79%	22.75%	24.00%	24.34%	24.89%	24.88%
Interval	17.73%	20.03%	21.24%	21.65%	21.45%	21.66%
Permutation	21.25%	23.33%	24.06%	25.75%	25.57%	25.77%
Simple IP-SEG	21.41%	22.90%	24.70%	24.91%	24.88%	25.79%
IP-SEG	22.77%	23.84%	25.99%	27.00%	26.63%	26.53%
M_1 - max reach | connected						
		1 visible	2 visible	3 visible	4 visible	5 visible
ERD		44.72%	60.61%	67.79%	69.98%	69.14%
GEO, $n = 1$		23.63%	29.58%	33.40%	34.56%	36.70%
GEO, $n = 2$		30.35%	43.38%	48.80%	51.89%	52.94%
GEO, $n = 3$		37.53%	51.82%	59.27%	59.36%	57.03%
Interval		29.46%	38.48%	46.75%	50.68%	50.14%
Permutation		36.11%	52.45%	60.59%	62.90%	58.91%
Simple IP-SEG		36.80%	52.95%	59.97%	61.87%	57.89%
IP-SEG		38.64%	54.82%	61.37%	61.82%	58.78%

Table 5.9: Consensus rates on dense ER graphs and on graph classes with geometric intersection models.

From Table 5.9 we can see that decentralized consensus on the randomly generated graphs with geometric intersection models is much more difficult to achieve than on equally dense ER graphs. This is true both in the baseline setting and and the setting with the best combination of visible nodes placement and model modification. The difference is particularly noticeable for unit interval (GEO, $n = 1$) and interval graphs. On the other hand, the consensus rates on classes such as simple IP-SEG and IP-SEG, whose geometric models contain permutation segments, are considerably closer to the consensus rates on ER graphs. We believe that this is related to the presence of a more pronounced community structure in random geometric and interval graphs. It is likely that substituting some

interval with permutation segments leads to additional connections between communities. Another takeaway is that even on networks on which consensus is more difficult to reach, we can attain a major improvement in the consensus rate of more than 15% by combining the increase of visible nodes, with optimizing their placement and increasing the trust that agents have in visible neighbors.

5.5 Conclusion

In this chapter, we considered decentralized consensus between agents positioned on nodes of a graph. We studied how the structure of graphs coming from different graph classes and random generating models can affect decentralized consensus, as well as the effectiveness of mechanisms intended to facilitate consensus. We reviewed our earlier experimental study in [111] on the value of communication in facilitating consensus which found that local communication can be of limited value, but may help when the underlying network contains high-degree hub nodes that transmit constrained, highly informative, messages. We then proceeded to perform an extensive simulations-based investigation of mechanisms for facilitating consensus in the presence of adversarial nodes, building on the work done in [56]. We considered special examples of graphs of different densities and found that adding more edges between nodes does not always help in reaching decentralized consensus and, in fact, may sometimes be detrimental. We showed that ensuring trusted nodes are placed on a connected subgraph can improve the chances of reaching a global consensus and that this improvement can be significantly amplified, if combined with increase in the trust that agents place in their visible neighbors. We explored different, relatively small, modifications to models of individual behavior, which also led to significant improvements in consensus rates. Based on this, we argued that strategies such as additional pre-play training and incentive modifications, can lead to better chances for consensus in experiments with human subjects. We also considered decentralized consensus on graphs with geometric intersection models, including random geometric graphs which have applications in ad hoc networks and robotic swarm systems. We found that reaching consensus on such graphs can be significantly more difficult, further illustrating the need for having mechanisms that could increase the chances for consensus.

There are a number of interesting questions for further study. On the simulations front, one may want to consider optimizing the placement of adversarial nodes in settings where visible nodes are connected. It is not obvious whether adversarial nodes would be better off if they are placed

so that they disconnect visible nodes from subsets of regular consensus nodes (structural impact) or maximize the number of neighbors they have (behavioral impact). In addition, it may be worthwhile exploring additional placements for visible nodes. We saw that consensus is particularly difficult to reach on networks such as random geometric graphs that exhibit community structure. In such settings, it is possible that a better way of facilitating consensus may be to maximize the number of inter-community edges covered by visible nodes, rather than maximize the reach of visible nodes. A related line of inquiry would be to explore the above questions on other random graph generating models which, in addition to community structure, capture other real-world network properties such as the Watts-Strogatz small-world model [112] or the geometric variant of the BA preferential attachment model due to Jordan [64].

On the modeling front, one possibility is to consider expanded models of individual behavior that would be appropriate for more complex settings that include varying individual incentives and communication. While modeling free-form communication would be quite challenging and may require much larger experiments with human subjects to collect enough training data, constrained communication, as described in [111], may be a more feasible starting point as modeling the individual communication behavior will deal primarily with frequency of communication, rather than content. Finally, on the experimental front, it would be valuable to further validate, in experiments with human subjects, the effectiveness of mechanisms for facilitating decentralized consensus, that have shown most promise in agent-based simulations.

BIBLIOGRAPHY

[1] ABBAS, W., LASZKA, A., AND KOUTSOUKOS, X. Improving network connectivity and robustness using trusted nodes with application to resilient consensus. *IEEE Transactions on Control of Network Systems 5*, 4 (2017), 2036–2048.

[2] ABBAS, W., VOROBEYCHIK, Y., AND KOUTSOUKOS, X. Resilient consensus protocol in the presence of trusted nodes. In *2014 7th International Symposium on Resilient Control Systems (ISRCS)* (2014), IEEE, pp. 1–7.

[3] ACAN, H., AND HITCZENKO, P. On random trees obtained from permutation graphs. *Discrete Mathematics 339*, 12 (2016), 2871–2883.

[4] ALEKSEEV, V. On the entropy values of hereditary classes of graphs. *Discrete Mathematics and Applications 3*, 2 (1993), 191–200.

[5] ALON, N., FELDMAN, M., LEV, O., AND TENNENHOLTZ, M. How robust is the wisdom of the crowds? In *Twenty-Fourth International Joint Conference on Artificial Intelligence* (2015).

[6] ANTONIONI, A., AND TOMASSINI, M. Degree correlations in random geometric graphs. *Physical Review E 86*, 3 (2012), 037101.

[7] BANDELT, H.-J., AND MULDER, H. M. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B 41*, 2 (1986), 182–208.

[8] BARABÁSI, A.-L., AND ALBERT, R. Emergence of scaling in random networks. *science 286*, 5439 (1999), 509–512.

[9] BARABÁSI, A.-L., ET AL. *Network science*. Cambridge university press, 2016.

[10] BENZER, S. On the topology of the genetic fine structure. *Proceedings of the National Academy of Sciences of the United States of America 45*, 11 (1959), 1607.

[11] BOCCALETTI, S., LATORA, V., MORENO, Y., CHAVEZ, M., AND HWANG, D.-U. Complex networks: Structure and dynamics. *Physics reports 424*, 4-5 (2006), 175–308.

[12] BONDY, J. A., AND MURTY, U. S. R. *Graph theory*. Springer, 2008.

[13] BOOTH, K. S., AND LUEKER, G. S. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *Journal of computer and system sciences 13*, 3 (1976), 335–379.

[14] BRACHA, G., AND TOUEG, S. Resilient consensus protocols. In *Proceedings of the second annual ACM symposium on Principles of distributed computing* (1983), pp. 12–26.

[15] BRANDSTÄDT, A., LE, V. B., AND SPINRAD, J. P. *Graph classes: a survey*. SIAM, 1999.

[16] BREU, H., AND KIRKPATRICK, D. G. Unit disk graph recognition is np-hard. *Computational Geometry 9*, 1-2 (1998), 3–24.

[17] CABELLO, S., CARDINAL, J., AND LANGERMAN, S. The clique problem in ray intersection graphs. *Discrete & computational geometry 50*, 3 (2013), 771–783.

[18] CHAKRABORTY, T., JUDD, S., KEARNS, M., AND TAN, J. A behavioral study of bargaining in social networks. In *Proceedings of the 11th ACM conference on Electronic commerce* (2010), pp. 243–252.

[19] CHANDOO, M. On the implicit graph conjecture. *arXiv preprint arXiv:1603.01977* (2016).

[20] CHATURVEDI, M. A parametric classification of directed acyclic graphs. Master's thesis, Colorado State University, 2017.

[21] CHVATAL, V. Perfectly ordered graphs. In *Topics on perfect graphs*, vol. 88. North-Holland Mathematics Studies, 1984, pp. 63–65.

[22] CLARK, B. N., COLBOURN, C. J., AND JOHNSON, D. S. Unit disk graphs. In *Annals of Discrete Mathematics*, vol. 48. Elsevier, 1991, pp. 165–177.

[23] CORNEIL, D. G., AND KAMULA, P. A. Extensions of permutation and interval graphs. In *Proc. 18th Southeastern Conference on Combinatorics, Graph Theory and Computing* (1987), pp. 267–276.

[24] CORNEIL, D. G., OLARIU, S., AND STEWART, L. The ultimate interval graph recognition algorithm? In *SODA* (1998), vol. 98, pp. 175–180.

[25] CRUCITTI, P., LATORA, V., AND MARCHIORI, M. Model for cascading failures in complex networks. *Physical Review E 69*, 4 (2004), 045104.

[26] DAGAN, I., GOLUMBIC, M. C., AND PINTER, R. Y. Trapezoid graphs and their coloring. *Discrete Applied Mathematics 21*, 1 (1988), 35–46.

[27] DAMASCHKE, P. Forbidden ordered subgraphs. In *Topics in combinatorics and graph theory*. Springer, 1990, pp. 219–229.

[28] DE RIDDER, H., ET AL. Information system on graph classes and their inclusions (isgci), 2020.

[29] DEMICHELIS, S., AND WEIBULL, J. W. Language, meaning, and games: A model of communication, coordination, and evolution. *American Economic Review 98*, 4 (2008), 1292–1311.

[30] DÍAZ, J., PETIT, J., AND SERNA, M. A survey of graph layout problems. *ACM Computing Surveys (CSUR) 34*, 3 (2002), 313–356.

[31] DIMOVSKA, M., AND MATERASSI, D. A control theoretic look at granger causality: extending topology reconstruction to networks with direct feedthroughs. *IEEE Transactions on Automatic Control* (2020).

[32] DIRAC, G. A. On rigid circuit graphs. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* (1961), vol. 25, Springer, pp. 71–76.

[33] DOLINSKAYA, I. S., SHI, Z. E., SMILOWITZ, K. R., AND ROSS, M. Decentralized approaches to logistics coordination in humanitarian relief. In *IIE Annual Conference. Proceedings* (2011), Institute of Industrial and Systems Engineers (IISE), p. 1.

[34] DUFFUS, D., GINN, M., AND RÖDL, V. On the computational complexity of ordered subgraph recognition. *Random Structures & Algorithms 7*, 3 (1995), 223–268.

[35] DUSHNIK, B., AND MILLER, E. W. Partially ordered sets. *American journal of mathematics 63*, 3 (1941), 600–610.

[36] ECKHOFF, J. Extremal interval graphs. *Journal of graph theory 17*, 1 (1993), 117–127.

[37] EHRLICH, G., EVEN, S., AND TARJAN, R. E. Intersection graphs of curves in the plane. *Journal of Combinatorial Theory, Series B 21*, 1 (1976), 8–20.

[38] ELLINGSEN, T., AND ÖSTLING, R. When does communication improve coordination? *American Economic Review 100*, 4 (2010), 1695–1724.

[39] ERDÖS, P., AND RÉNYI, A. On random graphs. *Publicationes Mathematicae (Debrecen) 6* (1959), 290–297.

[40] FEUILLOLEY, L., AND HABIB, M. Graph classes and forbidden patterns on three vertices. *arXiv preprint arXiv:1812.05913* (2018).

[41] FISHBURN, P. C. *Interval orders and interval graphs: A study of partially ordered sets*. John Wiley & Sons, 1985.

[42] FOWLER, C. A., RICHARDSON, M. J., MARSH, K. L., AND SHOCKLEY, K. D. Language use, coordination, and the emergence of cooperative action. In *Coordination: Neural, behavioral and social dynamics*. Springer, 2008, pp. 261–279.

[43] FRASER, B., COYLE, A., HUNJET, R., AND SZABO, C. An analytic latency model for a next-hop data-ferrying swarm on random geometric graphs. *IEEE Access 8* (2020), 48929–48942.

[44] GALLAI, T. Transitiv orientierbare graphen. *Acta Mathematica Hungarica 18*, 1-2 (1967), 25–66.

[45] GAREY, M. R., AND JOHNSON, D. S. *Computers and intractability*. W.H. Freeman and Company, 1979.

[46] GASHLER, M., AND MARTINEZ, T. Robust manifold learning with cyclecut. *Connection Science 24*, 1 (2012), 57–69.

[47] GAVRIL, F. Algorithms for maximum weight induced paths. *Information Processing Letters 81*, 4 (2002), 203–208.

[48] GERVACIO, S. V., RAPANUT, T. A., AND RAMOS, P. C. F. Characterization and construction of permutation graphs. *Open Journal of Discrete Mathematics 3* (2013), 33–38.

[49] GILES, A. P., GEORGIOU, O., AND DETTMANN, C. P. Connectivity of soft random geometric graphs over annuli. *Journal of Statistical Physics 162*, 4 (2016), 1068–1083.

[50] GOLUMBIC, M. C. *Algorithmic graph theory and perfect graphs*. Elsevier, 2004.

[51] GOLUMBIC, M. C., AND JAMISON, R. E. Edge and vertex intersection of paths in a tree. *Discrete Mathematics 55*, 2 (1985), 151–159.

[52] GOLUMBIC, M. C., TRENK, A. N., ET AL. *Tolerance graphs*, vol. 89. Cambridge University Press, 2004.

[53] GRACIA-LÁZARO, C., FERRER, A., RUIZ, G., TARANCÓN, A., CUESTA, J. A., SÁNCHEZ, A., AND MORENO, Y. Heterogeneous networks do not promote cooperation when humans play a prisoner's dilemma. *Proceedings of the National Academy of Sciences 109*, 32 (2012), 12922–12926.

[54] GRÖTSCHEL, M., LOVÁSZ, L., AND SCHRIJVER, A. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica 1*, 2 (1981), 169–197.

[55] HABIB, M., AND MOUATADID, L. Maximum induced matching algorithms via vertex ordering characterizations. *Algorithmica 82*, 2 (2020), 260–278.

[56] HAJAJ, C., YU, S., JOVESKI, Z., GUO, Y., AND VOROBEYCHIK, Y. Adversarial coordination on social networks. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems* (2019), pp. 1515–1523.

[57] HAJAJ, C., YU, S., JOVESKI, Z., AND VOROBEYCHIK, Y. Adversarial coordination on social networks. *arXiv preprint arXiv:1808.01173* (2018).

[58] HAMBURGER, P., MCCONNELL, R. M., PÓR, A., SPINRAD, J. P., AND XU, Z. Double threshold digraphs. In *43rd International Symposium on Mathematical Foundations of Computer Science* (2018), vol. 117, pp. 69:1–69:12.

[59] HARTMAN, I. B.-A., NEWMAN, I., AND ZIV, R. On grid intersection graphs. *Discrete Mathematics 87*, 1 (1991), 41–52.

[60] HELBING, D., BROCKMANN, D., CHADEFAUX, T., DONNAY, K., BLANKE, U., WOOLLEY-MEZA, O., MOUSSAID, M., JOHANSSON, A., KRAUSE, J., SCHUTTE, S., ET AL. Saving human lives: What complexity science and information systems can contribute. *Journal of statistical physics 158*, 3 (2015), 735–781.

[61] HELL, P., MOHAR, B., AND RAFIEY, A. Ordering without forbidden patterns. In *European Symposium on Algorithms* (2014), Springer, pp. 554–565.

[62] HOWORKA, E. A characterization of ptolemaic graphs. *Journal of Graph Theory 5*, 3 (1981), 323–331.

[63] ISHIZEKI, T., OTACHI, Y., AND YAMAZAKI, K. An improved algorithm for the longest induced path problem on k-chordal graphs. *Discrete applied mathematics 156*, 15 (2008), 3057–3059.

[64] JORDAN, J. Geometric preferential attachment in non-uniform metric spaces. *Electronic Journal of Probability 18* (2013).

[65] JOVESKI, Z., AND HAJAJ, C. Increasing the value of trusted nodes in adversarial decentralized networked consensus. *In preparation* (2020).

[66] JOVESKI, Z., AND SPINRAD, J. P. Some graph classes with two-property vertex orderings. In *The 6th Gdansk Workshop on Graph Theory* (2018), pp. 38–39.

[67] JOVESKI, Z., AND SPINRAD, J. P. Interval-permutation segment graphs. *To appear in Congressus Numerantium* (2019).

[68] JOVESKI, Z., AND SPINRAD, J. P. Algorithms for interval-permutation segment graphs. *In preparation* (2020).

[69] JUDD, S., KEARNS, M., AND VOROBEYCHIK, Y. Behavioral dynamics and influence in networked coloring and consensus. *Proceedings of the National Academy of Sciences 107*, 34 (2010), 14978–14982.

[70] KANN, V. Strong lower bounds on the approximability of some npo pb-complete maximization problems. In *International Symposium on Mathematical Foundations of Computer Science* (1995), Springer, pp. 227–236.

[71] KANNAN, S., NAOR, M., AND RUDICH, S. Implicit representation of graphs. *SIAM Journal on Discrete Mathematics 5*, 4 (1992), 596–603.

[72] KAY, D. C., AND CHARTRAND, G. A characterization of certain ptolemaic graphs. *Canadian Journal of Mathematics 17* (1965), 342–346.

[73] KEARNS, M., JUDD, S., TAN, J., AND WORTMAN, J. Behavioral experiments on biased voting in networks. *Proceedings of the National Academy of Sciences 106*, 5 (2009), 1347–1352.

[74] KEARNS, M., SURI, S., AND MONTFORT, N. An experimental study of the coloring problem on human subject networks. *science 313*, 5788 (2006), 824–827.

[75] KIMURA, M., AND SAITO, K. Tractable models for information diffusion in social networks. In *European conference on principles of data mining and knowledge discovery* (2006), Springer, pp. 259–271.

[76] KRATOCHVÍL, J., AND NEŠETŘIL, J. Independent set and clique problems in intersection-defined classes of graphs. *Commentationes Mathematicae Universitatis Carolinae 31*, 1 (1990), 85–93.

[77] KRATSCH, D. Finding the minimum bandwidth of an interval graph. *Information and Computation 74*, 2 (1987), 140–158.

[78] KUMAR, R., NOVAK, J., AND TOMKINS, A. Structure and evolution of online social networks. In *Link mining: models, algorithms, and applications*. Springer, 2010, pp. 337–357.

[79] LEBLANC, H. J., ZHANG, H., KOUTSOUKOS, X., AND SUNDARAM, S. Resilient asymptotic consensus in robust networks. *IEEE Journal on Selected Areas in Communications 31*, 4 (2013), 766–781.

[80] LIBEN-NOWELL, D., AND KLEINBERG, J. The link-prediction problem for social networks. *Journal of the American society for information science and technology 58*, 7 (2007), 1019–1031.

[81] MACPHAIL, L. H., NEUWIRTH, E. B., AND BELLOWS, J. Coordination of diabetes care in four delivery models using an electronic health record. *Medical care* (2009), 993–999.

[82] MAFFRAY, F. On the coloration of perfect graphs. In *Recent Advances in Algorithms and Combinatorics*. Springer, 2003, pp. 65–84.

[83] MAHESH, R., RANGAN, C. P., AND SRINIVASAN, A. On finding the minimum bandwidth of interval graphs. *Information and Computation 95*, 2 (1991), 218–224.

[84] MCCREERY, H. F., DIX, Z. A., BREED, M. D., AND NAGPAL, R. Collective strategy for obstacle navigation during cooperative transport by ants. *Journal of Experimental Biology 219*, 21 (2016), 3366–3375.

[85] MCCUBBINS, M. D., PATURI, R., AND WELLER, N. Connected coordination: Network structure and group coordination. *American Politics Research 37*, 5 (2009), 899–920.

[86] MIDDENDORF, M., AND PFEIFFER, F. The max clique problem in classes of string-graphs. *Discrete mathematics 108*, 1-3 (1992), 365–372.

[87] MONIZ, H., NEVES, N. F., AND CORREIA, M. Turquois: Byzantine consensus in wireless ad hoc networks. In *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)* (2010), IEEE, pp. 537–546.

[88] NEOGI, R., RAMAN, V., AND FRANCIS, M. Recognizing k-clique extendible orderings. In *International Workshop on Graph-Theoretic Concepts in Computer Science* (2020), Springer.

[89] NEWMAN, M. E. Spread of epidemic disease on networks. *Physical review E 66*, 1 (2002), 016128.

[90] NEWMAN, M. E. The structure and function of complex networks. *SIAM review 45*, 2 (2003), 167–256.

[91] OLARIU, S. All variations on perfectly orderable graphs. *Journal of Combinatorial Theory, Series B 45*, 2 (1988), 150–159.

[92] ORMAN, K., LABATUT, V., AND CHERIFI, H. An empirical study of the relation between community structure and transitivity. In *Complex Networks*. Springer, 2013, pp. 99–110.

[93] PEARL, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[94] PUJARA, J., MIAO, H., GETOOR, L., AND COHEN, W. Knowledge graph identification. In *International Semantic Web Conference* (2013), Springer, pp. 542–557.

[95] RAGHAVAN, V., AND SPINRAD, J. Robust algorithms for restricted domains. *Journal of algorithms 48*, 1 (2003), 160–172.

[96] RAMALINGAM, G., AND RANGAN, C. P. A unified approach to domination problems on interval graphs. *Information Processing Letters 27*, 5 (1988), 271–274.

[97] REID, C. R., LUTZ, M. J., POWELL, S., KAO, A. B., COUZIN, I. D., AND GARNIER, S. Army ants dynamically adjust living bridges in response to a cost–benefit trade-off. *Proceedings of the National Academy of Sciences 112*, 49 (2015), 15113–15118.

[98] REN, W., BEARD, R. W., AND ATKINS, E. M. A survey of consensus problems in multi-agent coordination. In *Proceedings of the 2005, American Control Conference, 2005.* (2005), IEEE, pp. 1859–1864.

[99] ROTMENSCH, M., HALPERN, Y., TLIMAT, A., HORNG, S., AND SONTAG, D. Learning a health knowledge graph from electronic medical records. *Scientific reports 7*, 1 (2017), 1–11.

[100] SARCEVIC, A., PALEN, L., WHITE, J., STARBIRD, K., BAGDOURI, M., AND ANDERSON, K. " beacons of hope" in decentralized coordination: learning from on-the-ground medical twitterers during the 2010 haiti earthquake. In *Proceedings of the ACM 2012 conference on computer supported cooperative work* (2012), pp. 47–56.

[101] SCHEINERMAN, E. R. Random interval graphs. *Combinatorica 8*, 4 (1988), 357–371.

[102] SCHEINERMAN, E. R. An evolution of interval graphs. *Discrete Mathematics 82*, 3 (1990), 287–302.

[103] SCHEINERMAN, E. R., AND ZITO, J. On the size of hereditary classes of graphs. *Journal of Combinatorial Theory, Series B 61*, 1 (1994), 16–39.

[104] SPINRAD, J. P. *Efficient graph representations*. American Mathematical Society, 2003.

[105] SZÁMADÓ, S. Pre-hunt communication provides context for the evolution of early human language. *Biological Theory 5*, 4 (2010), 366–382.

[106] SZKLARCZYK, D., FRANCESCHINI, A., WYDER, S., FORSLUND, K., HELLER, D., HUERTA-CEPAS, J., SIMONOVIC, M., ROTH, A., SANTOS, A., TSAFOU, K. P., ET AL. String v10: protein–protein interaction networks, integrated over the tree of life. *Nucleic acids research 43*, D1 (2015), D447–D452.

[107] TAKAOKA, A. A vertex ordering characterization of simple-triangle graphs. *Discrete Mathematics 341*, 12 (2018), 3281–3287.

[108] TRENCHARD, H., AND PERC, M. Energy saving mechanisms, collective behavior and the variation range hypothesis in biological systems: a review. *Biosystems 147* (2016), 40–66.

[109] USEVITCH, J., AND PANAGOU, D. Resilient leader-follower consensus to arbitrary reference values. In *2018 Annual American Control Conference (ACC)* (2018), IEEE, pp. 1292–1298.

[110] VERMA, T., AND PEARL, J. Causal networks: Semantics and expressiveness. In *Machine intelligence and pattern recognition*, vol. 9. Elsevier, 1990, pp. 69–76.

[111] VOROBEYCHIK, Y., JOVESKI, Z., AND YU, S. Does communication help people coordinate? *PLOS ONE 12*, 2 (2017).

[112] WATTS, D. J., AND STROGATZ, S. H. Collective dynamics of 'small-world'networks. *nature 393*, 6684 (1998), 440–442.

[113] ZHANG, H., VOROBEYCHIK, Y., LETCHFORD, J., AND LAKKARAJU, K. Data-driven agent-based modeling, with application to rooftop solar adoption. *Autonomous Agents and Multi-Agent Systems 30*, 6 (2016), 1023–1049.