An Introduction to Algorithmic Graph Theory

Part 1

Martin Charles Golumbic

In this introduction, I will start off by telling you how I see algorithmic graph theory. I’ll do this in the context of a number of areas related to my research work for over four dozen years. This will be a gentle introduction—we’ll come back to most of these subjects as the semester progresses.

First, I will describe, or remind many of you of, the notion of intersection graphs, and specifically, the family of interval graphs. We will learn something about their structure and apply this to the topic of greedy coloring of interval graphs. Then I’ll tell you a story, called the Berge mystery story, in case you have not heard it before. We'll move on to other structured families of graphs, and the delicious topic of graph sandwich problems, which is a good theme before lunch. We’ll finish the introduction by mentioning the notions of probe graphs and tolerance graphs.

Some Basics

Let’s start off with some very basic ideas that most undergraduates will know. It will also serve to establish the terminology and notation needed further on.

A graph is a collection of vertices and edges, with each edge connecting some pair of vertices referred to as its endpoints. These are often called simple, undirected graphs. Two vertices that are connected by an edge are called adjacent to each other.

Properly coloring a graph is the process of giving a color to every vertex so that adjacent vertices have different colors. Figure 1.1 shows a graph that is colored using three different colors, and, of course, all three colors are required because you have a triangle—the triangle forces you to have three colors. So this is an example of an optimal coloring.

An independent set, sometimes called a stable set, is a collection of vertices, no two of which are connected by an edge. In our example, the set of vertices colored grey is a stable set; also, the three bottom vertices \{d, e, f\} form another independent set.

A clique, or a complete set, is a collection of vertices, every two of which are connected by an edge. A triangle, like \{a, b, d\} in our example, is a clique of size 3, and an edge, like\{c, e\} is a clique of size 2. Cliques and independent sets are just the opposite of each other, they are complimentary notions.
The complement of a graph is formed by exchanging the edges for non-edges. We usually denote the complement of a graph $G=(V,E)$ by putting a bar over it, that is, $\overline{G}=(V,\overline{E})$ where $\overline{E}=$\{uv $|$ u,v $\in$ V, uv $\notin$ E\}. In our example, the complement of the my original graph is shown in Figure 1.2. I've rearranged the vertices in drawing it so it looks nicer laid out on the plane. Here you can see here that ed is an edge in $G_{bar}$ whereas originally e and d were independent in G.

![Figure 1.1 The graph G.](image1)

![Figure 1.2 The graph $G_{bar}$.](image2)

All of this is probably familiar to you, as well as the notion of a directed graph. A directed graph (or digraph) is defined similarly, but where you put a direction on each edge. We usually call these arcs as opposed to edges. You may have one arc directed from vertex a to vertex b and another directed from b to a, or you might just have a single arc going from one vertex to another, as illustrated in Figure 1.3(a). If you erase the directions on the arcs, we call the result its underlying undirected graph, as illustrated in Figure 1.3(b).

![Figure 1.3 (a) A directed graph, and (b) its underlying undirected graph.](image3)

Now there is a special kind of directed graph which we call an orientation. An orientation is a digraph where you have only one direction per edge. It's like assigning one way streets! Looking at Figure 1.4, you see two different orientations of the pentagon. The first one is cyclic, meaning you can go around and around in the cycle forever, like “Charlie on the M.T.A.”. The second is acyclic, meaning there is no endless cycle, so Charlie’s fate in this case will always be “learn’d”—arrested for not having an extra nickel.

In general, for any digraph $D$, we will define it as cyclic if it contains some cyclically oriented cycle. If $D$ has no such cyclically oriented cycle, we often call it a dag (directed acyclic graph). In a dag $D$, there will
always be at least one source (a vertex with no incoming arcs) and there will always be at least one sink (a vertex with no outgoing arcs). Exercise.

Figure 1.4. A cyclic orientation and an acyclic orientation of the pentagon.

Intersection Graphs and Interval Graphs

The concept of an intersection graph appears in many applications in computer science, engineering, biology, operations research and artificial intelligence. It also provides a rich source of mathematical properties as you'll see throughout the course. This symbiosis, going forth and back between applications and the mathematical structure that lies behind them, will be a constant theme.

I would like to relate some graph problems to applications. You may have seen problems involving intervals on a line, like the scheduling example illustrated in Figure 1.5. Here you have to reason about a set of time intervals. Let’s define the intersection graph of intervals on the line. We create a vertex for every interval, 1, 2, 3, 4, 5, and create an edge between two vertices in the graph if their associated intervals intersect. Thus, in our example, Phase 1 and Task 4 intersect, so I put an edge in between 1 and 4. Phase 2 and Phase 1 don't intersect; I've drawn them so that they just don't quite touch. There is no edge between 1 and 2, etc.

Figure 1.5. A set of intervals on a line and its intersection graph.

The theory of interval graphs, the intersection graphs of intervals, is an important area of study in structural graph theory. And it goes back a long way. To even before I was born; well not quite. There was a Hungarian mathematician named Hajos who in 1957 wrote a very short problem in the problems section of a German language mathematics journal. He asked the question, “What sort of graphs can be obtained by considering the intersection of intervals?” as in this scheduling problem. Thus, he introduced the problem of characterizing interval graphs.

Around the same time, an American biologist by the name of Seymour Benzer was looking at problems of genetic structure, specifically studying the structure of overlapping pieces of genetic material. In his 1959
paper, he hypothesized that these pieces were linear segments, and he too asked about the combinatorial structure of intersecting of intervals on the line, and how they would overlap. Hajos was looking at time intervals; Benzer was looking at gene strings. Today we might think more in terms of pattern matching or pattern intersection of intervals.

About five years past, and Gilmore and Hoffman from IBM, Guila Houri, a student of Claude Berge, and Lekkerkerker and Boland gave the first mathematical characterizations of interval graphs. Then 12 years later, the first really efficient linear time algorithm to recognize this family of graphs appeared in the literature, given by Booth and Lueker. There are many other applications that come up along the way, and keep coming up. Applications to radio frequency assignment, to VLSI design, temporal reasoning in AI, storage allocation, and the list goes on and on.

I used to refer to something that I called The Endless Lemma, which stated that every time you think you know every application of interval graphs, someone will point out another one to you. That was at least 25 years ago, but I think it still applies. It was motivated back then when I heard about an application for analyzing pavement deterioration: you had roads that were deteriorating along certain stretches (intervals) that were made out of different materials, and the paper analyzed how these different materials being used to build the highways were aging and deteriorating over time. So interval graphs have quite a lot of history, and it is amazing that there are still interesting questions about them today.

About a year and a half after I finished my doctorate at Columbia University, when I was starting to write my book Algorithmic Graph Theory and Perfect Graphs, I wanted to find the original problem as written by Hajos. All the time during my studies, I simply couldn't find this journal anywhere in New York City. How is that possible? I don't know. But sometime after I'd graduated while attending a math conference in Helsinki, I spent some hours at the library at the University of Helsinki and found the journal. Copying out the page by hand carefully in German, and then translating it, I finally had the source in time to quote from it in my book.

Let’s now look at a few applications of interval graphs. Here is a typical easy room scheduling example. There are number of lectures that need to be assigned classrooms at the university. Lecture $a$ may go from 9:00 to 10:15, Lecture $b$ may go from 10:00 to 12:00, etc. Conflicting lectures require different rooms; that is, when two time intervals overlap, we cannot schedule those lectures to the same room, as is the case for Lectures $a$ and $b$. Question: How many rooms do I need to schedule the full set of lectures?
Figure 1.6. A scheduling example [from Golumbic and Trenk, 2004].

Figure 1.6 illustrates such a scheduling problem with six lectures. The vertical cut at 13:15 indicates a time when four classes are “alive” at the same moment, implying that we will need at least four classrooms. But what happens if I only have three classrooms? We’ll get to that in a minute. Meanwhile, look at the diagram in terms of its interval graph, i.e., the intersection graph of the intervals. Figure 1.7 shows the interval graph and its complement.

![Diagram](https://via.placeholder.com/150)

Figure 1.7. (a) The interval graph and (b) its oriented complement.

What is the complement of intersection? **Disjointness.** Disjointness has a very special property on the line, namely, if two intervals are disjoint, then we always know that one interval is before the other. And that implies a natural direction to each edge of the complement, from the later to the earlier, making it an acyclic digraph, as in Figure 1.7(b).

Some of you may recognize that the scheduling problem of assigning classrooms to lectures is just a simple example of graph coloring on its interval graph. Each classroom is a color; adjacent vertices correspond to intersecting intervals which must receive different colors/rooms. Now every student of computer science knows that graph coloring is usually a computationally hard problem to solve optimally—it is an NP-complete problem, in general. But interval graphs have special structural properties that allow us to color them efficiently!
The very efficient algorithm for coloring a set of intervals is by a greedy sweep across the representation from left to right assigning colors in a first fit fashion. We will prove that this greedy algorithm uses the optimal (smallest possible) number of colors. For example, Figure 1.8 shows a diagram of eight intervals drawn on a line below it. Underneath, I have drawn them again in a way that may remind you of a circuit routing problem, where one must connect vertical pins with non-overlapping horizontal electrical wires (yet another real-world application). The coloring shown in the figure is the optimal greedy coloring, constructed as follows:

From left to right, the first interval encountered is a, and we will give it the first color “solid red”. The second interval encountered is b, giving it the next color “dashed black”. Interval c is next and is assigned the color “dotted turquoise”. But look at what happens next—interval b finishes, so b's color becomes available again. Thus, when we now come to interval d the greedy algorithm will reuse that color, namely, “dashed black” being assigned to d. Moving right along, a finishes so “solid red” is available, thus, interval e opens at pin 7, and we assign it color “solid red”.

Now we have come to the point where interval f opens, and all three colors that have been used so far are “busy”. So we have to open up a fourth color “dashy-dotted bluish-purple” assigning it to f. Continuing this way, releasing a color for further use when an interval finishes, and assigning the lowest available free color when the next interval opens, we obtain a coloring of all the intervals where intersecting intervals have different colors.

In our example, we have used four colors. If you had done a greedy sweep from right to left, you would have gotten a different coloring, but it still would have used four colors.

In the lower diagram of Figure 1.8, we connect a set of matched pairs of pins, where (i) horizontal lines intersecting vertical lines don’t bother me (I can have a little bridge there, or put them on different layers in the plane), but horizontals need exclusive use of their levels. In circuit design, these levels are often called lanes in a street. Just as minimizing classrooms in our scheduling example translated into minimizing the number of colors, here too minimizing the number of levels in this connecting diagram translates into interval graph coloring.
Is Greedy the best we can do? Can we prove that it's optimal? The answer is yes—it does use the smallest number of colors, and I’d like to go through a proof of it with you.

Let $k$ be the number of colors used by the greedy algorithm on a set of intervals $I$. We want to show that $k$ is the best possible. Let’s look at the point $P$ on the line when the $k$th color was used for the first time. The reason for having to open up this color $k$ was that at $P$ all of the other colors, 1, 2, …, $k-1$ were busy, and we were forced to use a new color $k$. Therefore, at that very point $P$, you see that there are $k$ intersecting intervals, and these form clique in the interval graph of size $k$. Thus, any coloring solution will require at least $k$ colors—a clique of size $k$ must use at least $k$ colors. This proves optimality.

In our example, there is the clique of size four, and Greedy successfully painted the intervals with four colors. Formally, we would say that, on the one hand, at least $k$ colors are required because of the clique, and on the other hand this greedy method succeeded in using $k$ colors, since that’s how $k$ was defined, so therefore the solution is optimal.
Characterizing Interval Graphs

The other day, I was walking around campus when someone stopped me to show me a graph. “I’d like a nice fast algorithm that would look at this graph and say, ‘A ha! an interval graph’ or ‘Oh ho! not an interval graph’.” “Sure,” I replied, “I’ve got just what you need back in my office. But first, I’d like you to know something about their mathematical structure.”

There are the two important properties that together characterize interval graphs; they go back 1964 and the work of Gilmore and Hoffman. One is called the chordal graph property and the other is the co-TRO property. I will describe them next.

A graph is called a **chordal graph** if every cycle of length greater than or equal to 4 has a **chord**, that is, an edge connecting two vertices that are not consecutive in the cycle. In other words, chordal graphs may not contain chordless cycles. For example, a pentagon is a chordless cycle of length 5, because it is missing of the possible diagonal shortcuts in it. A four cycle is not chordal, since it is missing a chord either between two opposite vertices. In modern terminology, these chordless cycles are sometimes call **holes**. The French used to call a graph that has no holes a **triangulated graph**, ‘graph triangulaire’, which is what I called them in my first book because I was very French in those days. Another old name for them was **rigid circuit graph**. But the terminology has evolved and most people, including me, now call them chordal graphs.

Interval graphs turn out to have this property, that is, **interval graphs are chordal graphs**—they may not contain chordless cycles. Why?

Well it's fairly simple to see this; you can even explain this to your high school sister. If I were trying to represent the chordless 6-cycle with a set of intervals on the line, I would get into trouble.

Let’s start with vertex b and its two neighbors a and c which are not connected. Wherever the interval for b is placed on my line, the intervals for c and a must hit b but may not hit each other. We can arbitrarily put a first hitting b and then it stopping with a little gap covered by b, and then c begins, and so forth. Now we add d to it, using the same argument: c has already hit b but it must also hit d, however has to be disjoint from b, so b ends, then d begins. Similarly, when c ends, e may start when d ends, f may start. But now I've got trouble—we cannot closing the cycle. If f has to hit a, there's no way to do it, since we are on the line. No way to just circumvent this problem. And this is the whole proof of the fact that interval graphs are chordal. Here I've shown it for six cycle but it is the same proof for all
lengths chordless cycles and for any graph that contains a hole.

![Figure 1.8. The interval graph cannot contain a chordless cycle.](image)

To understand the second property that I want to present, we need to know about the Transitive Orientation problem. We have already seen what an orientation is, namely, where you assign one way directions to every edge of a graph. A **transitive orientation** is an orientation which, in addition, obeys the transitive property: if an edge ab is oriented from vertex a towards vertex b, and edge bc is oriented from b to c, then there is an edge ac and it is oriented from a to c. Formally, we say that an orientation $F$ is transitive if $ab, bc \in F \implies ac \in F$.

In Figure 1.9, we have what I call the A graph because it looks like the letter A. The first orientation (a) is not transitive, since we have a path $d \rightarrow b \rightarrow a$ but $da$ is not an edge. So this is not a transitive orientation. However, the second orientation (b) is a transitive orientation—there is only one oriented path $b \rightarrow a \rightarrow c$ and it does have the required shortcut $b \rightarrow c$. The difference between these two examples can be seen by a kind of “forcing” relation between the edges. An orientation of the edge $a \rightarrow c$ forces the direction $e \rightarrow c$ because if it went the other way, there would be a violation of transitivity. This was the problem with orientation (a) not being transitive. We'll see some more of that forcing later on in Chapter 5.

![Figure 1.9. (a) A non-transitive orientation and (b) a transitive orientation of the graph A.](image)

We call a graph **transitively orientable (TRO)** if it admits an orientation which is transitive. The A-graph is TRO, but the pentagon is not TRO. Finally, we say that a graph is **co-TRO** if it is the complement of a TRO graph. I will claim here that interval graphs are co-TRO, that is, the complement of an interval graph has a transitive orientation. Why?

I hinted to this before. We saw that an edge in the complement of an interval graph represents **disjointness** between the two associated intervals. So if we orient this edge from the earlier interval to the
later interval, this will be a transitive orientation/relation. Namely, if in the complement I have interval a being disjoint and to the right of interval b, and interval b being disjoint and to the right of interval c, then we must have interval a being disjoint and to the right of interval c. Thus, orienting the edges of the complement in that direction from the earlier to the later, then this orientation will be transitive. And if the complement is TRO, then the interval graph is co-TRO.

Combining these two properties, give the following characterization proven first by Gilmore and Hoffman 1964:

A graph is an interval graph if and only if G is chordal and its complement is transitively orientable.

We have just seen one direction of the proof. For the proof of the opposite direction, I would ask that you read the proof in Chapter 5 of AGT&PG.

Now back to the fellow who stopped me on campus. This characterization of interval graphs provides the basis for the first generation of recognition algorithms for the class of interval graphs, in the early 1970s. Researchers came up with an algorithm to test whether a graph is chordal, and another algorithm to test whether a graph is transitively orientable. Putting those two algorithms together, as phase 1 and phase 2, the first algorithm to recognize interval graphs was born. In fact, if you go back to the paper by Gilmore and Hoffman there is an implicit algorithm that is somewhat different, but no one actually looked at that so much as an algorithm, and certainly in ’64 people weren't thinking of things in terms of complexity. They didn't even care too much about polynomial complexity vs. exponential complexity until about 1969 or so.

Gilmore and Hoffman both worked at IBM Yorktown Heights; Gilmore also taught a course in logic as an adjunct professor at Columbia, and I took that course. I don't know if Hoffman ever taught a course there, but he did get his doctorate from Columbia.

Now I am going to tell you the Berge Mystery Story. There are various versions, and here is my version. Claude Berge, who authored the original version in the ‘60s eventually in 1994 wrote a murder mystery based on this mathematical theorem: *Qui a tué le duc de Densmore?* *(Who killed the Duke of Densmore? Bibliothèque Oulipienne No.67)* How many of you read French? There is a new book that came out in the last year *L’AGRAPHEUR*, written by Alain Hertz. The book is collection of original stories about a graph theorist detective. It is a very cute book, especially fun because he used a lot of French puns in it. The English and German translations are a good approximation.

**The Berge Mystery Story**

Six professors had been to the library on the day that the rare tractate was stolen. Each had entered once, stayed for some time and then left. If two were in the library at the same time, then at least one of them saw the other. Detectives questioned the professors, because they are trying to track down who stole the tractate, the volume, and they collected the following testimony.
Abe said that he saw Burt and Eddie
Burt reported that he saw Abe and Ida
Charlotte claimed to have seen Desmond and Ida
Desmond said that he saw Abe and Ida
Eddie testified to seeing Burt and Charlotte
Ida said that she saw Charlotte and Eddie

But one of the professors lied, who was it?

In Figure 1.3(a), the set of facts is illustrated in the form of a digraph, where each directed edge indicated a statement of “fact”: who said they saw whom. Now, double arrows immediately imply “truth”, for example, Burt said he saw Abe and Abe said he saw Burt, so even if one of them was lying and the other was telling the truth, so they really were in the library at the same time. In the case of Ida and Eddie, however, maybe they were in the library or maybe not depending on whether Ida is the liar. So in Figure 1.3(b), I have drawn a bold line to show immediate truth, that there is definitely an intersection of the intervals in which they were in the library, and a fine line to indicate possible intersection.

What I actually do now, before showing my solution, is to walk out of the room and leave everyone to take a piece of paper and start doodling--to figure out who is the liar in the story. So please stop reading, and start doodling.

Here is my explanation. You will notice a chordless cycle in Figure 1.3(b). The cycle A,B,I,D implies that one these professors is the liar, since if all of these guys were telling the truth, the whole truth and nothing but the truth, we would have a contradiction to the property that an interval graph has no chordless cycle! This observation will help to solve the problem: Since Charlotte and Eddie do not participate in the cycle A,B,I,D, they must have told the truth, because the liar is among the four in the cycle. But that’s not the only cycle around here, for example, there is the big chordless cycle going all the way around A,E,C,D, implying that Burt and Ida have also told the truth. So now we are down to two possible liars, Abe or Desmond. We can keep them locked up for extra questioning, and send everyone else home.

Missing remainder of argument.................

Now this is just a toy example in some sense, was Desmond stupid or just ignorant? Well, if Desmond has studied algorithmic graph theory he would have known that his testimony to the police would not hold up, so he would have done something else. He would have lied differently.
Other Families of Intersection Graphs

In 1969, Victor Klee wrote a short paper titled, “What are the intersection graphs of arcs in a circle?”. In this four page paper, he basically said, let's look at a circle and a set of arcs on that circle, just as we have done for intervals on a line. We can ask all the same questions about the intersection graph of these arcs in a circle. Figure 2.1 shows some arcs in a circle and its intersection graph, defined in the same way: Every circular arc has a vertex; two vertices are adjacent if those arcs intersect.

Circular arc graphs are different from interval graphs, most notably since they may contain chordless cycles, like g,b,c,e,f,g in our example. The whole reason why cycles didn't work for intervals, but it work fine for circular arcs is that the host is a circle and not a line. In this model, people can schedule not only interval events, but also periodic events—events that repeat themselves, like the time that a traffic light is red in your driving lane.

Formally, given a collection of subsets \{S_1, S_2, \ldots, S_n\} of a set S, we define its intersection graph G=(V,E) as follows: V={1, 2, \ldots, n} and, for any pair i,j \in V, we have

\[ i\cap j \in E \text{ if and only if } S_i \cap S_j \neq \emptyset \]

The host set S might be the real line, a tree, 3-dimensional Euclidean space, the program flow diagram of a computer program, a grid on the plane, and the subsets might be of a certain type, like intervals on the line, paths or subtrees of the tree, rectilinear boxes in 3-space, live areas of the variables in a computer program, closed curves on a grid, etc.

In fact, I see Klee's paper as an implicit challenge to consider a whole variety of problems, on many kinds of intersection graphs. Enough to build a whole course worth of lectures. You could talk about the intersection graph of boxes in the plane, or paths in the tree, or chords in the circle, or spheres in 3-space, or trapezoids, parallelograms, curves of functions, and many other geometrical and topological bodies—enough to keep a person like me busy for a whole career. Even a whole bunch of people, for their whole career—provided, that there was something interesting to say about the structure of the graph class.

Moreover, from an applications point of view, you might want to know, for example, about properties of intersecting boxes on the plane if you're laying out packages on a floor, or if you're trying to assign areas
for projects or machinery in a manufacturing plant. And then the algorithmic problems that come up are especially important—how to recognize that you have this particular kind of graph, how to color them, how to find maximum cliques or maximum independent sets, and other algorithmic problems applied to such special families of graphs.

Here is an example of a hierarchy of four types of intersection diagrams, each giving rise to a corresponding family of intersection graphs. I'll start at the bottom. The first diagram is called a permutation diagram. We put the numbers 1 through \( n \) on the upper line (in this example \( n=5 \)) and some permutation on the lower line; we connect the matching numbers with straight line segments. This is called a permutation diagram. Some of those segments cross, and some of them don’t cross. If they cross they intersect, and I will put an edge in the associated intersection graph. The class of graphs that arise as the intersection graphs of permutation diagrams are called permutation graphs.

Here is an example of a hierarchy of four types of intersection diagrams, each giving rise to a corresponding family of intersection graphs. I’ll start at the bottom. The first diagram is called a permutation diagram. We put the numbers 1 through \( n \) on the upper line (in this example \( n=5 \)) and some permutation on the lower line; we connect the matching numbers with straight line segments. This is called a permutation diagram. Some of those segments cross, and some of them don’t cross. If they cross they intersect, and I will put an edge in the associated intersection graph. The class of graphs that arise as the intersection graphs of permutation diagrams are called permutation graphs.

![Image](image-url)

Figure 2.2. A small hierarchy of intersection graphs (from Golubic & Trenk, 2004).

Now supposed we make the diagram a little bit more general. Instead of having segments that connect numbers, let’s change the numbers into little intervals on the upper and lower lines, where matching intervals have the same length, thus forming a set of parallelograms. This type of diagram, the second from the bottom, is called a parallelogram diagram and its intersection graph we will call a parallelogram graph.

We can generalize this further by making the intervals in the upper and lower lines have different lengths, so the bodies become trapezoids, and the intersection graphs of this type of trapezoid diagram are called trapezoid graphs. Finally, we can allow these bodies to have curly lines as their side boundaries, instead of with straight lines, making them look like twisted ribbons. You might call them ribbon graphs, but I don’t think you’ll find that in the literature, because I made it up. But what you will find is that these are equivalent to the diagrams known as function diagrams, resulting in function graphs. Function diagrams are very much like the permutation diagrams, except instead of intersecting straight segments of
permutations, here I am allowed curves--curves that are function curves, meaning that once you start going from the top to the bottom, you can't go back--they are monotone going down.

All four of these graph classes are different. With some trial and error, you can find a graph that is a trapezoid graph, but fails to be a parallelogram graph, or a graph that is a parallelogram graph, but fails to be a permutation graph. Remarkably, this top graph class turns out to be equivalent to the co-comparability graphs, that is, the co-TRO graphs. Comparability and transitive orientation are two different names for the same concept. We just saw the co-TRO property in interval graphs. In fact, the intersection graph of the function diagrams are exactly the co-TRO graphs. We’ll see other properties related to this, but for this introduction, I just wanted to show an example of this kind of intersection graph.

After a short break, we will continue with another variation of intersection graphs, where we play with the size of the intersection, sometimes ignoring or tolerating small intersections.