An Investigation of Monotone Properties of Graphs

Max Denning, Ben Hinthorne, Mackenzie Kong-Sivert, and Celine Park

7 May 2019

1 Abstract

This paper will focus on understanding monotone properties of graphs and their connection to evasiveness and testability. The first portion of this paper will begin by giving relevant background information and definitions in order to clearly define a monotone property of a graph and what problems have been explored concerning monotone properties. We will then discuss a few open problems in search of the maximum number of edges that can be packed into a graph on a fixed number of vertices such that the graph maintains its given monotone property. Specifically, we will include discussions of the monotone properties C_4-free and K_r-Free. We will then explore the evasiveness conjecture and its relation to monotone properties of graphs. We will explore how to determine if a property of a graph is evasive, and the relationship between monotone and evasive properties. Lastly, we will give an overview of what it means for a graph property to be testable and give a basic understanding of theorems about the testability of monotone properties.

2 Background

Before moving forward with any investigations, it is first important to understand the formal definition of what a monotone property of a graph is. To begin this understanding, we take an even further step back and define formally a property of a graph.

Definition 2.1: A *property* of a graph G is an aspect of a G that is closed under isomorphism. In other words, it is an aspect that is inherent to the structure G and does not depend on the labeling or representation of G. A property that applies to no graphs or to all graphs is considered *trivial* [2].

Defining a generic property of a graph allows us to move into defining *monotone* properties of a graph. There are a two different yet related definitions of monotone, which are defined as follows:

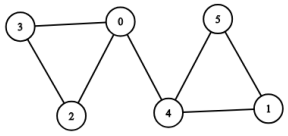
Definition 2.2: A property P is *monotone decreasing* in G if for every graph G for which P holds true, P must also be true for every subgraph of G [2].

Definition 2.3: A property P is *monotone increasing* in G if the addition of edges to G does not destroy property P [2].

The nuances of these two definitions are best understood by considering some examples of graph properties. Consider a graph G that has $n > 4$ vertices and has the property of being C_4-free. Figure 1 demonstrates that the property of being C_4-Free is monotone decreasing, but not monotone increasing.

For our investigations into placing a bound the maximum amount of edges we can add to a graph such that it maintains its given monotone property, we will be concerning ourselves with monotone decreasing properties. It is clear this bound for monotone increasing properties is trivial. Since we know that the monotone increasing property is closed under the addition of edges, we can continue to add edges until the graph is complete and know that the property will not be destroyed. In our discussions of evasiveness and testability, the theorems we present use the monotone decreasing definition as well. Therefore, for the remainder of this paper unless otherwise stated, when

Figure 1: This graph is C_4-Free, and will continue to be C_4-Free upon the removal of any vertex or edge. Notice, however, it will not continue to be C_4-Free on the addition of edges. Adding an edge between 0 and 5, for example, creates a C_4.



discussing monotone properties we will be referring to monotone decreasing properties.

The latter part of this paper will focus on explorations of evasive properties of graphs. We will define the key terminology needed here, and return to these definitions later as needed.

Definition 2.4: The *complexity* of a property P of a graph G is the number of vertex pairs that must be examined in order to determine if G has property P. Equivalently, this value is also the number of entries in the adjacency matrix of G that must be inspected to determine if P is a property of G [3].

It is important to note that a property may have multiple complexities since there may be multiple sets of vertex pairs that can be looked at to determine if G has property P. In this paper, we will largely concern ourselves with the *minimum complexity* of P, denoted $c(P)$, which is the smallest number of all complexities of a property P in G. From our definition of complexity, we can understand how a property P of a graph is evasive.

Definition 2.5: A property is said to be *evasive* if any process has to examine every possible pair of vertices in a graph in order to determine

if the graph has this property. In other words, if the $c(P) = n(n-1)$, where n is the number of vertices in G [3].

We will return to this definition later with a more in depth discussion of evasiveness. The final concept we will discuss is testability. The concept of testability relies heavily on complexity theory, and therefore much of our discussion will be based around defining concepts of complexity theory. Rather than giving those definitions here, we will leave these concepts for a more in depth discussion in a later section of the paper.

3 Open Cases of Monotone Properties

We will now look into some open cases of monotone properties. Specifically we will look at the graph properties C_4 - free and K_r - free (for $r \geq 4$). We will discuss whether these properties are monotone, and explore how many edges we can pack into a graph on a fixed number of vertices such that it maintains its given monotone property[1].

3.1 C_4 - free

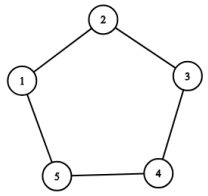
A graph with the property of C_4 - free is a graph with that contains no cycle of length of 4. As we mentioned earlier, this property is monotone in regards to the monotone decreasing definition. We now turn our attention to the question of how many edges we can pack into a graph that is satisfies the property of being C_4 - free on a fixed number of vertices such that the graph continues to be C_4 - free.

To illustrate this question further, let's see how many edges we can pack into a graph on 5 vertices such that it remains C_4 - free. We show in Figure 2 that we can construct a graph G on 5 vertices with 5 edges such that G is C_4 - free. Furthermore, if we were to add any other edge to this construction, G would then contain a C_4 subgraph and no longer be C_4 - free. It may seem, then, that this is the maximum number of edges we can pack into a graph on 5 vertices such that it is C_4 - free.

Figure 3, however, shows that in another configuration we can create a graph on 5 vertices and 6 edges that contains no C_4 subgraph. It turns out that this is the maximum number of edges that you can pack into a graph

[1]Note this is very similar to asking the extremal number of a property

Figure 2: This is a graph on 5 vertices and 5 edges that contains no C_4 such that adding any edge will create a C_4.



on 5 vertices such that the graph is C_4 - free, as any graph on 5 vertices with greater than 6 edges will contain a C_4. This example exhibits the nuance of the question that we are asking, showing that the configuration of edges within the graph can effect whether or not more edges can be added such that the graph continues to satisfy its monotone property.

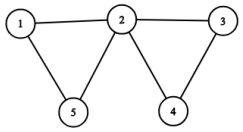
What remains open for the property of being C_4 - free is the amount of edges you can pack in a C_4 - free graph on some arbitrary number of edges. There has been much previous research into this problem, from which the main result is a theorem presented by Reimann.

Theorem 3.1: If a graph G is on n vertices and contains no C_4, then

$$|E| \leq \tfrac{n}{4}(1 + \sqrt{4n-3}).$$

Note that this theorem presents an upper bound on the number of edges in graph that is C_4 free. We therefore know that if a graph on n vertices has more than this many edges, then there must exist a C_4. Equality of this bound, however, has not been proven. In other words, it remains open whether or not you can construct a graph that has exactly $\frac{n}{4}(1 + \sqrt{4n-3})$ edges. Further discussion of this concept is beyond the scope of this paper, and we will instead turn our attention to examples concerning other monotone properties.

Figure 3: This is a graph on 5 vertices and 6 edges that contains no C_4 such that adding any edge will create a C_4.



3.2 K_r - free

We see that property of K_r - free (specifically when $r \geq 4$) is monotone by the first definition of a monotone graph property. A graph that is K_r - free, means that it does not contain a subgraph with r vertices in which each vertex is adjacent to every other vertex in the subgraph. If a graph is K_r - free, then removing edges will not change the fact that the graph is K_r - free. This is because the graph is already lacking the appropriate amount of edges to have a K_r subgraph and removing edges will not change that fact. Thus, the graph property of K_r - free is a monotone graph property by our monotone decreasing definition. However, this property does not meet the conditions of our second definition of a monotone graph property. We can show this through a proof by contradiction. Let us assume, that K_r - free is a monotone property under our second definition. We have a graph with 4 vertices and 5 edges that is K_4 - free, if we add one more edge, the graph would then contain a graph a K_4 subgraph and no longer be K4 - free. Thus, the property is only monotone under our first definition of a monotone graph property and not the second.

We now present the question of how many edges we can pack into a K_r - free graph such that it remains K_r - free. A similar bound to that presented by Riemann for C_4 - free graphs is presented by Mantle specifically for K_3 - free graphs.

Theorem 3.2: If a graph G is on n vertices and contains no K_3, then

$$|E| \leq \tfrac{n^2}{4}.$$

Unlike Theorem 3.1, equality of the expression given in Theorem 3.2 has been proven when G contains an even number of vertices. We will now turn our attention to one last example of a monotone property.

3.3 K-colorable

Similarly to the property of K_r-free, $k-$colorability (for $k \geq 3$) is monotone under the first definition of a monotone graph property. If a graph G is $k-$colorable for some $k \geq 3$, then it implies that any pair of adjacent vertices in G are assigned two distinct colors. Any subgraph of G, then is also $k-$colorable, by keeping the original color assignments from G itself.

However, drawing from the second definition of a monotone graph property, we can pose this as an extremal number question. K-colorability is akin to partitioning the vertices in a graph into k disjoint sets, such that the graph is k-partite, meaning that no two vertices u, v such that $u, v \in S_i$ are adjacent, where S_i denotes one such disjoint set. A complete k-partite graph on n vertices is called a Turan graph, notated as $T_{n,k}$, and for this subset of k-partite (and thus k-colorable) graphs, there has been prior work done on finding the extremal number. It was proven that the number of edges, $t_{n,k} = |E(T_{n,k})|$ is equal to

$$\binom{r}{2}\left\lceil\frac{n}{k}\right\rceil^2 + r(k-r)\left\lceil\frac{n}{k}\right\rceil\left\lfloor\frac{n}{k}\right\rfloor + \binom{k-r}{2}\left\lfloor\frac{n}{k}\right\rfloor^2,$$

where r is the number of sets in the graph with size exactly $\left\lceil\frac{n}{k}\right\rceil$. It was also proven that $\mathrm{ex}(n, K_r) = t_{n,r1}$, and so for complete $k-$partite graphs on n vertices, $\mathrm{ex}(n, T_{n,k}) = \mathrm{ex}(n, K_{r+1})$ [5]. This then transforms the $k-$colorability problem into a version of the extremal number problem for K_r graphs.

4 Evasiveness Conjecture

4.1 The Aanderaa-Rosenberg Conjecture

One of the early landmark discoveries in the field of evasiveness came in 1975 with the paper "A generalization and proof of the Aanderaa-Rosenberg conjecture" by Ronald Rivest and Jean Vuillemin. Years earlier, Ronold L. Rosenberg had conjectured that, for any nontrivial graph property P and a graph G on v vertices that is represented as an adjacency matrix, any algorithm that determines whether G has property P must make $\Omega(v^2)$ inspections of the matrix for G in the worst case. Stål Aanderaa disproved this conjecture with a counterexample[2], but he proposed instead a modified version of the conjecture. Specifically, he restricted the set of properties to only *monotone* properties. [4] Thus, we have the Aanderaa-Rosenberg Conjecture:

Conjecture (Aanderaa-Rosenberg): For any nontrivial, monotone graph property P and a graph G on v vertices that is represented as an adjacency matrix, any algorithm that determines whether G has property P must make $\Omega(v^2)$ inspections of the matrix for G in the worst case.

This is the conjecture that was proven by Rivest and Vuillemin. Many proofs in complexity theory papers rely on oracles, which is a construction in which the authors suppose for the sake of the proof that there is a magical oracle that knows the answer to some problem and can be queried in constant time. The authors of this paper point out that their proof does not rely on oracles. Rather, they suppose that $C(P) < d$, where d is the number of arguments in the boolean function for determining P, and work based on the conditions necessary for that to occur.

4.2 The Aanderaa-Karp-Rosenberg Conjecture

With the Aanderaa-Rosenberg conjecture proven, Richard Karp expanded this conjecture to state that the set of properties to which the Aanderaa-Rosenberg conjecture applies can also be shown to be evasive. This conjecture

[2]In particular, he showed that fewer than $3v$ inspections are needed to determine whether a directed graph on v vertices contains a vertex with in-degree $v-1$ and out-degree 0.

is also called the evasiveness conjecture. This conjecture has never been proven, although significant work has been done to tackle it.

Conjecture (Aanderaa-Karp-Rosenberg): Any nontrivial, monotone graph property P and a graph G on v vertices that is represented as an adjacency matrix must be evasive.

Some of the main theorems that have been proven about the evasiveness conjecture are elaborated on below.

4.2.1 Graphs with a Prime Power Number of Vertices

Although this conjecture has not been formally proven, there are specific cases that have been proven. For example, in 1984 Jeff Kahn, Michael Saks, and Dean Surtevant proved the evasiveness conjecture for graphs whose number of vertices is a prime power. They accomplish by leveraging some topological definitions:

Definition: An (abstract simplicial) *complex* Δ of a finite set X is a collection of subsets of X such that $A \subseteq B \in \Delta$ implies that $A \in \Delta$. [3]

The astute reader will notice that this bears a striking structural resemblance to the definition of monotone graph properties. This is a particularly interesting example because it uses the structures of another mathematical discipline to achieve a result in graph theory. The proof is rather involved, and we do not have the space to explain it here, but the main result is as follows:

Theorem: Any nontrivial monotone property of graphs (or digraphs) on v vertices has complexity at least $v^2/4 + o(v^2)$.

It is then shown that, as a corollary, the evasiveness conjecture holds for graphs whose number of vertices is a prime power.

4.2.2 Bipartite Graph Properties

Later on, in 1988, Andrew Chi-Chih Yao proved that the evasiveness conjecture applies for all bipartite graph properties. As one might assume, a bipartite graph property is a property that applies specifically to bipartite graphs. Like Kahn, Saks, and Surtevant, Yao uses topological definitions to

prove this, citing their 1984 paper [6]. This means that these three people not only leveraged another sub-discipline for a graph theoretic result, they also created a precedent for others to do the same. This practice can be found in many other papers on the evasiveness conjecture, but we do not have time to mention all of them.

Yao takes this idea further, however, by incorporating another topological concept, the string property.

Definition: A *string property* P is a function from $0, 1^t$ into $0, 1$. That is, it is a function from a binary string to a single binary value. [6]

With this definition, Yao goes on to put the decision tree for a bipartite graph property into a string property in order to prove the above result.

4.3 Examples of Evasiveness

Empty Graph The graph property of emptiness (there are no edges in the graph) is evasive. When telling whether a graph is evasive or not, we need to look at the worst case. The worst case is, essentially, the maximum amount of edges one would need to check in order to tell if a graph contains a specific property or not. The worst case for an empty graph can occur if a graph is empty or contains only 1 edge. If the graph is empty, we would need to check every possible edge in order to ensure there that no edges exist. If the graph does contain a single edge, there is a chance that one could check all the potential edges except that edge, before checking this edge. We can see an example of both worst cases in Figure 4. If we checked all the edges in the order of the numeric value of their vertices, we would need to check all the edges in the empty graph on the left since no edges exist to disprove it. It is therefore clear that in the worst case we would need to check every edge in order to determine if G is empty, signifying that the property of emptiness is evasive.

Contains a Cycle of length 3 The graph property of having a cycle of 3 (a triangle), is also an evasive property. This is because the worst case of verifying this property occurs when we have checked all the edges for triangles except one edges and both vertices on that potential edge are adjacent to the same vertex. The remaining edge must be checked in order to confirm whether or not a triangle exists. We can see an

Figure 4: Empty graph and graph with single edge

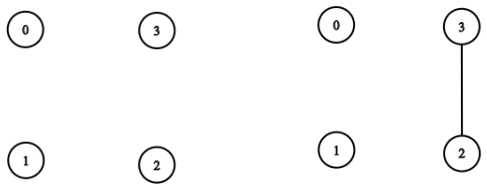
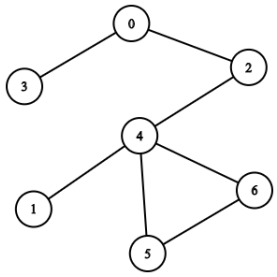


Figure 5: Graph with a 3 cycle subgraph



example of this case in Figure 5. In this graph, if one were to search for triangles in the graph starting if the vertices if the lowest numeric label, the last triangle that would be checked is the triangle between vertices 4, 5, and 6. Since there are no other 3 cycles in the graph, all the other 3 cycle combinations would have been tested. Thus, all 3 cycle combinations would need to be tested in order to verify whether a cycle existed or not. Once again, it is therefore clear that in the worst case we would need to query every edge in order to determine if the graph contains a cycle of length 3, indicating that this property is evasive.

5 Testability

Having now given an overview of evasiveness and its connections to monotone properties of graphs, we will introduce the concept of testability to further our understanding of monotone graph properties and their applications. Much of the known research connecting monotone properties and testability involves advanced complexity theory that is beyond the scope of this paper. Without the space to fully explore these topics, we will instead give the reader an intuitive understanding of the basics of testability. The goal of this section is for the reader to understand intuitively what it means for a graph property to be testable so they may begin to understand how testability relates to monotone properties. We begin by giving the definition of a tester for a graph property, and will continue to elaborate on this definition.

Definition 5.1: A *tester* for a graph property P is randomized algorithm, which given a graph G, the number of vertices (we will call this n), and the ability to query whether a desired pair of vertices in G are adjacent or not distinguishes with high probability (often times $\frac{2}{3}$) between G satisfying property P and being ϵ-far from satisfying P [1].

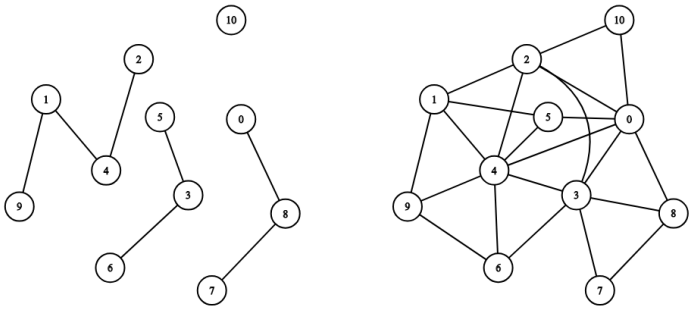
To fully understand this definition, we first must understand what it means for a graph to be ϵ-far from satisfying a property P.

Definition 5.2: A graph G is said to be *ϵ-far* from satisfying property P if one must add or delete at least ϵn^2 edges in order to turn G into a graph satisfying P [1].

There are a few nuances of this definition worth exploring further. First note that the maximum number of edges a graph can have is $\binom{n}{2}$, and it therefore follows that ϵ must be less than 1 since a graph cannot contain n^2 edges. Furthermore, note that for a graph that is ϵ-far from satisfying P, the larger ϵ is the more vertices one would need to add or delete in the graph in order to satisfy property P. Therefore, the magnitude of the ϵ you choose for your tester for property P is proportional to "how far away" you are checking that G is from satisfying property P.

For example, if I choose ϵ to be 2/3, my tester will report if a graph G is satisfying property P or if there needs to be at least $2n^2/3$ edges added or removed for G to satisfy P. This is a significant amount of edges needed to be added or removed for G to satisfy P, and therefore in this case if my

Figure 6: Imagine we build a tester for the property P which is that that a graph contains no edges. We chose $\epsilon = \frac{1}{20}$. For the two graphs below, notice that both graphs need at least ϵn^2 edges removed to satisfy property P, yet one is obviously much closer to satisfying P than the other.



tester reported that G is ϵ-far from satisfying P, I would know that G is no where close to satisfying P.

However, if I chose ϵ to be a much smaller value, say something like 1/20, then my tester would check if G is satisfying property P or if at least $n^2/20$ edges would be needed to be added for removed in order G to satisfy property P. For cases in which ϵ is very small we can understand that the phrase "at least" in the definition of ϵ-far is quite important. Figure 6 gives an example of how when ϵ is small, reporting that G is ϵ-far from satisfying a property can give us very little information about how far G actually is from satisfying the property.

This example clearly shows that the choice of epsilon can have a great impact on the specificity of the results of your tester. The significance of the choice of a value for epsilon is even further highlighted when we understand what it means for a graph property to be *testable*.

Definition 5.3: A property P is said to be *testable* if the total number of queries needed for a tester on property P is bounded only by ϵ and is therefore independent of the number of vertices in G [1]. In other words, a property P is testable if you can build a tester for it such that the run-time of the tester is bounded only by your choice of ϵ and is

entirely independent of the size of the graph.

We can see how the choice of epsilon is even more significant for testable properties. Choosing epsilon for testable properties not only determines how specific the results of the tester could be, but also determines the computational complexity of the tester.

From our understanding of testers and what it means for a property of a graph to be testable, we can now give a surprising result connecting monotone properties of graphs and testability.

Theorem 5.1: Every monotone graph property is testable[3] [1].

The result of this theorem is quite significant. Recalling our understanding of what a testable property is, this theorem signifies that we can build a tester for any monotone property P such that the run time is bounded only by ϵ and is not at all dependent on the size of the graph. Thus no matter how large a graph gets, if the property we are concerned about is monotone, determining between G satisfying property P and being ϵ-far from satisfying P is dependent only on the choice of ϵ.

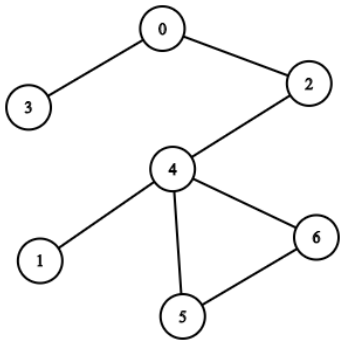
The proof of this theorem relies heavily on complexity theory beyond the scope of this paper. Rather than giving this proof, we will try to give a conceptual understanding of the theorem.

Let's say we are trying to build a tester for a monotone property and we chose our ϵ to be $\frac{1}{4}$. Thus the question we are asking is: for a graph on n vertices, does G satisfy property P or do we need to add or remove at least $\frac{n^2}{4}$ edges in order for G to satisfy property P. First consider a case when G is on 10 vertices, a relatively small number of vertices. We thus ask does G satisfy property P, or do we need to add or remove at least 25 edges for G to satisfy property P. Now consider a second case when G is on 1000 vertices. We are then asking whether G satisfies P or if we need to add or remove at least 250,000 edges in order for G to satisfy P.

Although the adding or removing 250,000 edges to a graph is much greater than adding or removing 25 edges, notice that the graph on more vertices has many more potential edges that could be added or removed. In fact, no matter the size of the two graphs, the tester is actually always asking the same question about how many edges need to be added or removed *proportional*

[3]It has actually been shown that every monotone graph property is testable with *one-sided error*, but we will not explore this concept in this paper.

Figure 7: Recall the figure from earlier in the paper of a graph G that contains only one 3-cycle.



to how many edges could possibly be added or removed. In addition, this proportion is entirely dependent on ϵ. With this notion that the question the tester is asking can be viewed as entirely dependent on the choice of ϵ, we can gain an understanding for why the run-time of the tester would also be only dependent on ϵ

The significance of Theorem 5.1 is even further understood when recalling that every monotone property is evasive. Therefore, because monotone properties are evasive, determining whether or not a graph G satisfies such a property can be very computationally expensive when the size of the graph gets very large. Since every monotone property is testable, however, we can attempt to ask a slightly different question that may not give as concise of an answer, but is much less computationally expensive. There are many cases in which it would be valuable to ask whether an evasive property is testable because asking if it simply exists in a graph or not can be expensive. There are some cases worth mentioning, however, where testability will not save us from the evasive problem.

To illustrate one of these cases we return to the example of the monotone property of not containing any 3-cycles. Recall the scenario in which a graph G contains only one 3-cycle, as demonstrated in 7 and thus determining if G does not contains a 3-cycle could be quite expensive as the size of G grows. In this case, though, only one edge would be needed to removed for G to now be free of any 3-cycle. Therefore, for any ϵ we chose such that $\epsilon n^2 > 1$[4], our

[4]note that if we chose an ϵ such that $\epsilon n^2 < 1$, we are asking a very similar question to

tester will not be able to determine if G is ϵ-far from satisfying the desired property or not because only one edge needs to be removed, and therefore reporting that ¿1 edge need be removed would be false.

From this example it is clear that although monotone properties being testable can be helpful in certain cases, there are limitations to what a tester for a monotone property can report.

6 Conclusion

We began this paper by introducing the concept of monotone properties of graphs. We gave basic definitions and built our understanding of monotone properties through a few simple examples. We continued our exploration of monotone properties by exploring how many edges we can pack into a graph such that the graph will continue to satisfy its monotone property. We then moved into applications and extensions of monotone properties, by introducing the concept of evasiveness. We discussed the progression of literature on the topic of evasiveness, moving from the Aanderaa-Rosenberg Conjecture to the revised Aanderaa-Karp-Rosenberg Conjecture. Having understood these conjectures, we gave some examples of showing how monotone properties are evasive. Finally, we introduced some basics of complexity theory to understand what it means for a property to be testable. From this knowledge, we were able to gain an intuitive understanding of why every monotone property is testable. Further explorations of these topics require a more in depth understanding of topology and complexity theory. In conclusion, we hope that this paper gave the reader an intuitive understanding of monotone properties of graphs and their applications, so they may now set out to further explore these concepts.

the evasive question: if G simply contains property P or not.

References

[1] Noga Alon and Asaf Shapira. Every monotone graph property is testable. *SIAM Journal on Computing*, 38(2):128–137, 2005.

[2] Nesetril-Jaroslav Graham, Ronald and Steve Butler. *The Mathematics of Paul Erdos II*. 2013.

[3] Jeff Kahn, Michael Saks, and Dean Sturtevant. A topological approach to evasiveness. *Combinatorica*, 4(4):297–306, 1984.

[4] Ronald L Rivest and Jean Vuillemin. A generalization and proof of the aanderaa-rosenberg conjecture. In *Proceedings of the seventh annual ACM symposium on Theory of computing*, pages 6–11. ACM, 1975.

[5] P. Turan. On an external problem in graph theory. *Mat. Fiz. Lapok*, 48:436–452, 1941.

[6] Andrew Chi-Chih Yao. Monotone bipartite graph properties are evasive. *SIAM Journal on Computing*, 17(3):517–520, 1988.

Appendix A Complexity Classes

Since many of the results explained in this paper have to do with the asymptotic complexity of certain problems, it may be useful to explain the significance of some of these terms. First, we will explain some basic terms to do with complexity classes. For any given process or calculation, we can give a general statement expressing how many steps it will take to complete, as a function of the size of the input. The size of this input is denoted n. The asymptotic complexity of this process is such a statement.

Here, only the general shape of the function is important. If the function is multiplied by a constant, the constant is not taken into consideration. Likewise, if the function is expressed as the sum of multiple terms, only the term with the fastest growth is expressed. The reason for this is in the name "asymptotic complexity"; for sufficiently large n, constant coefficients and smaller added terms will be overshadowed by the growth pattern of the function.

For example, it is well-known that for a list of size n, printing every pair of items in that list will take $n(n-1)/2$ steps. Since multiplying by a constant is not important for our purposes, we can instead express this as $n(n-1)$. We, however, are still not done; $n(n-1)$ can be expressed as $n^2 - n$. Because $n < n^2$ (for sufficiently large n), we can say that the whole asymptotic complexity of this process is n^2.

A.1 Big-O Notation

Not many processes are as simple to describe as the example above, and for some processes, the amount of time it takes to complete depends heavily on the configuration of the problem. For example, suppose you were searching through a list (that is in no particular order) for a desired item. If that item happens to be the first or second item in the list, the asymptotic complexity of your search has been 1. On the other hand, if the desired item is the last item in the list, you will have to look through all n items find it, so the asymptotic complexity of your search will be n. What, then, is the asymptotic complexity of your search? This depends on which type of complexity you use.

Perhaps the most common way of measuring asymptotic complexity is what is commonly called "big-O" notation. Let ALG denote some problem whose complexity we wish to express. We say that ALG $\in O(g(n))$ for some function $g(n)$ if for all cases, the complexity of ALG is at most $g(n)$.

A.2 More Notation

Another form of notation that is frequently used when talking about the complexity of functions is Ω, or "big-omega" notation. A function $f(n)$ is said to be in $\Omega(g(n))$ when there exists some (real) constant c and integer n_0 such that $n \geq n_0$ implies $f(n) \geq cg(n)$. That is, for a large enough n, $f(n)$ is at least $g(n)$, up to a constant. This is a sort of lower bound, as opposed to $O(g(n))$, which is a sort of upper bound on the growth of $f(n)$.