Graph Decomposition and NP-Completeness

Senior project for B.A. in Mathematics

Benjamin Gustafson

Trinity University

Advised by Brian Miceli, PhD

May 2023

1 Introduction

This project is concerned with the computational complexity of some problems related to graph decompositions. An H-decomposition of a graph G is a partition of the edges of G into subgraphs isomorphic to H. Some problems related to graph decomposition are known to be NP-complete. If we fix a graph H, where H contains a connected component with three edges or more, the problem of determining whether a given graph G admits an H-decomposition is NP-complete [DT97].

We propose the following problem that we suspect is NP-complete: Given a graph G on n vertices and a positive integer m, is there a G-decomposition of mK_n (the m-fold complete multigraph on n vertices)? The problem is clearly in NP, since if we are given a solution we need only check that each edge of K_n is covered exactly m times. This will take polynomial time since there are polynomially many edges in K_n and mK_n.

We will show that a similar problem that takes a multigraph as input is NP-complete. We can show that the problem is NP-hard by a reduction from a modified version of the partition problem.

There is not an obvious polynomial time algorithm for this problem. When solving the problem, an algorithm must face the choice of how to arrange the copies of G. It seems as though we are forced into trying multiple options for these copies, which will lead to branching and an exponential time algorithm.

We present an algorithm for solving the problem, and some results from that algorithm. The algorithm is able to solve instances where the number of vertices is less than 6 and m is less than 5. Above that, the runtime becomes intractable.

2 Graph Decomposition

An H-decomposition of a graph $G = (V, E)$ is a partition of E into subgraphs isomorphic to H. For example, the complete graph on 5 vertices, K_5, can be decomposed into two copies of the 5-cycle, C_5 1.

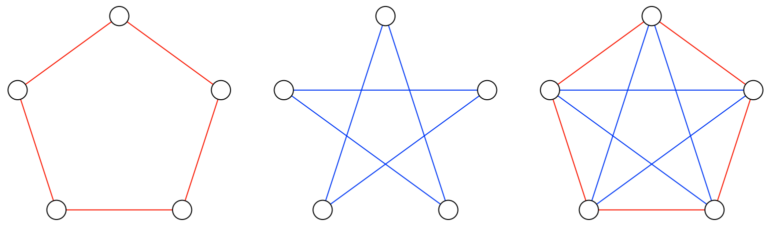


Figure 1: A C_5-decomposition of K_5

Wilson [Wil75] showed that for sufficiently large n, there exists an H-decomposition of K_n (the complete graph on n vertices) if and only if the number of edges of H divides the number of edges of K_n, and the gcd of the vertex degrees of G divides the degree of the vertices of K_n.

Cameron and Cioabă [CC15] looked at the problem of decompositions of mK_n (the complete multigraph with m of each edge). For example, $2K_4$ can be decomposed into copies of the 3-star 2. They asked if the same divisibility conditions as Wilson's were sufficient for the existence of an They define the set $M(G) = \{m : mK_n \text{ can be partitioned into copies of } G\}$, and show that for any graph G, there is a positive integer m_0 and a finite set F of multiples of m_0 such that $M(G) = m_0\mathbb{N} - F$. They call this m_0 the partition modulus.

They define m_1 to be the value for which the divisibility conditions are equivalent to the assertion that $m_1|m$ for all $m \in M(G)$. Then if the divisibility conditions were sufficient for the existence

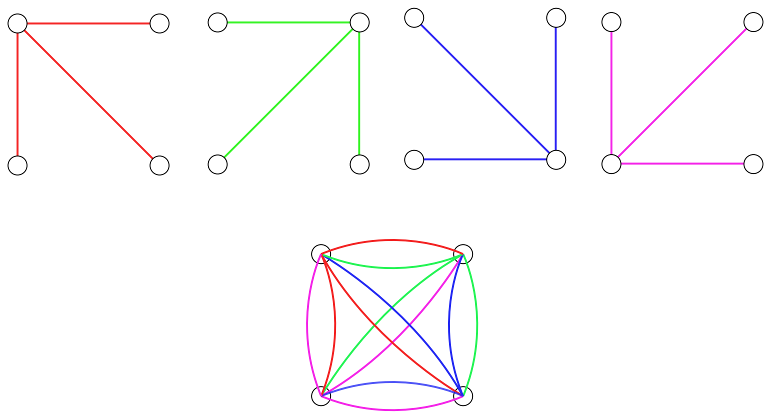


Figure 2: The 3-star decomposes $2K_4$

of an H-decomposition of mK_n for sufficiently large m, we would have $m_0 = m_1$. But, they found that there are graphs for which $m_0 \neq m_1$. Their counterexample is the n-star, where n is odd. It remains open whether there exists a simple way to determine the partition modulus.

Holyer [Hol81] showed that it is NP-complete to determine whether a given graph can be decomposed into isomorphic copies of K_n, for $n \geq 3$. Dor aand Tarsi [DT97] further showed that for a fixed graph H, where H contains a connected component with three edges or more, the problem of determining whether a given graph G admits an H-decomposition is NP-complete. The proof is a reduction from 3-dimension matching where each edge in the 3-dimensional matching instance corresponds to an edge in the graph G. They construct graphs that they call modules that can be decomposed by H to either include or not include a specific edge. These modules are then combined into the graph G such that each module encodes a choice of including or not including a edge in the 3-dimensional matching problem.

This result implies that the most general formulation of graph decomposition as a decision problem, where a graph G and a graph H are given and we want to determine if G admits an H-decomposition, is NP-complete. What is not implied to be NP-complete is the problem where H and G are both given, but G is restricted to a family of graphs. We consider this problem when the family of graphs is complete multigraphs, mK_n.

3 NP-Completeness

Complexity theory is the field of theoretical computer science that is concerned with how efficiently algorithms can run. Time complexity is a measure of how many operations an algorithm takes in terms of the size of the input. We say that an algorithm has time complexity $O(f(n))$ if there exists $N \in \mathbb{N}$ and $c > 0$ such that for any $n > N$ running the algorithm on an input of size n takes less than $cf(n)$ operations.

The time complexity of a problem is the time complexity of the fastest algorithm for solving the problem. We limit our consideration to decision problems, that is, problems for which the answer is either true or false. We are particularly interested in the difference between polynomial and exponential time complexity. The class of problems P is defined to be all problems with time complexity bounded above by some polynomial.

For example, consider the problem of searching an unordered list. We phrase the problem as a decision problem: given a unordered list of n integers, determine if a given integer k is in the list. In order to determine whether k is in the list, we must check each element until either we find k and stop, or k is not in the list. In the case where k is not in the list, we must do n operations. Thus, the problem has time complexity $O(n)$, and is in P.

A problem P is reducible to a problem Q if for each instance of P there exists an instance of Q such that a solution to P exists if and only if a solution to Q exists. We say that P is polynomial-time reducible to Q if the instance of Q can be generated from a given instance of P in polynomial time. If P is known to be polynomial-time reducible to Q, then if a polynomial time solution to Q exists, there exists a polynomial time solution to P. So we can say Q is harder than P in the sense that time complexity of Q is an upper bound for the time complexity of P, up to a polynomial. If P reduces to Q and Q reduces to P, then the two problems are as hard as each other, up to a polynomial. We say that the two problems are polynomially equivalent. For example, all problems in P are polynomially equivalent.

A problem is in the class nondeterministic polynomial time (NP) if its solutions can be checked in polynomial time. A problem is NP-complete if every problem in NP is polynomial-time reducible to it.

It is not known whether NP-complete problems have polynomial time solutions. If one NP-complete problem had a polynomial time solution, then every NP problem would be in P as well, since NP-complete problems are by definition the hardest problems in NP. So, if this were the case, we would have P = NP. Since P vs. NP is a well-known open problem, proving that a problem is NP-complete means that there is not a polynomial time solution by our current knowledge.

3.1 Proving NP-Completeness

The first problem proved to be NP-complete was the Boolean satisfiability problem (SAT). The input for SAT is a Boolean expression in conjunctive normal form. Conjunctive normal form (CNF) is an expression of the form $(x_{11} \vee x_{12} \vee \cdots x_{1k}) \wedge \cdots \wedge (x_{m1} \vee x_{m2} \vee \cdots x_{mk})$ where each x_{ij} is a literal. We are asked whether there is some assignment of the literals to true or false that makes the expression true. SAT was proved to be NP-complete by Cook [Coo71] by showing that any problem in NP can be reduced to SAT.

In order to prove that a problem Q is NP-complete we must show that (1) Q is a member of NP, and (2) there is an NP-complete problem that reduces in polynomial time to Q. It is almost always trivial to show that a problem is in NP, but it is often nontrivial to reduce the problem to a known NP-complete problem. While there is a reduction from every NP-complete problem to another, the choice of which problem to reduce from can greatly affect how complicated it is to make the reduction.

3.2 NP-Intermediate

For an NP problem that is not known to be in P or NP-complete, there is also the possibility that it is in a third class called NP-intermediate. In the case where P $\neq$ NP, there is guaranteed to be a class of NP problems that are neither P nor NP-complete. The problem of graph isomorphism is thought to be NP-intermediate. Determining if graph isomorphism is in P or NP-complete is a well known open problem.

For our problem of decomposing the complete graph into copies of a given graph, we know that it is NP, so it must be in at least one of P, NP-complete, or NP-intermediate. If we could show that the problem is polynomially equivalent to the graph isomorphism problem, then we would know that we will not be able to easily show that it is in P or NP-complete. However, there is not an obvious reduction from graph decomposition to graph isomorphism.

4 Complete Multigraph Decomposition is NP-Complete

We define the **complete multigraph decomposition problem** to be: given a multigraph G on n vertices and a positive integer m, determine if there exists a G-decomposition of mK_n (the complete graph on n vertices with multiplicity m).

The **partition problem** is: given a multiset S of positive integers, does there exist partition of S into two parts of equal sum. The partition problem is known to be NP-complete [GJ79]. For our purposes we need a modified version of this problem.

The **restricted partition problem** has as input a set S of n positive integers with the promise that n is even and there does not exist a multiset M of $n/2$ elements of S such that $\sum_{m\in M} m = \frac{1}{2}\sum_{s\in S} s$. The problem asks whether there exists a subset $S' \subset S$ of size $n/2$ such that $\sum_{s'\in S'} s' = \frac{1}{2}\sum_{s\in S} s$.

For example, if we are given $S = \{1, 2, 3, 4\}$ then $n = 4$ and the sum of the elements is 10. The subset $S' = \{1, 4\}$ is a solution since it has $n/2 = 2$ elements and sums to 5. For an example that fails the multiset condition, consider $S = \{1, 2, 3, 4, 5, 7\}$ which has $n = 6$ elements and sums to 22. The subset $S' = \{1, 3, 7\}$ has 3 elements and sums to 11, but there also exists the multiset $\{3, 3, 5\}$ with three elements that sums to 11.

We can show that the restricted partition problem is NP-complete by modifying the reduction from 3-dimensional matching (3DM) to the partition problem. For the reduction from 3DM to partition see Garey and Johnson [GJ79].

The **3-dimensional matching** problem is: given three sets W, X, Y with $|W| = |X| = |Y| = q$, and a set $W \subseteq W \times X \times Y$, does there exist a subset $M' \subseteq M$ such that each element of each of W, X, Y occurs in exactly one element of M'.

Lemma 1. *Restricted partition is NP-complete.*

Proof. Let the sets W, X, Y with $|W| = |X| = |Y| = q$, and $M \subseteq M \times X \times Y$ be a given instance of 3DM. We construct a set A of positive integers by specifying their binary representation. We divide the binary string into $3q$ zones, with each zone labeled by an element of $W \cup X \cup Y$. For each $m = (w, x, y) \in M$, we construct the integer $a \in A$ to have a 1 in the rightmost bit of the zones corresponding to w, x, y.

Choosing elements from the 3DM instance will correspond to choosing integers in the partition instance to reach the target sum. So, the zones need to have enough bits so that when we add up all of the elements in A we never carry from one zone to the next. Thus, each zone gets $p = \lceil \log_2(q+1) \rceil$ bits.

Then we define the integer T to have a 1 in the rightmost bit of all zones, and $S = \sum_{a \in A} a$. The set $A \cup \{S, T\}$ is the usual reduction to the partition problem. There is a solution to the instance of 3DM if and only if there is a partition of $A \cup \{S, T\}$ into two parts of equal sum.

We need to introduce more integers to satisfy the condition that the solution to the restricted partition problem has exactly half of the given elements. Let n_0 be sufficiently large so that $2S + T$ is less than n_0 bits. Then let b_i^1 be the binary string 1 followed by $n_0 + 3i$ zeros where $0 \leq i < |A| + 2$. Similarly let b_i^2 and b_i^3 be the binary strings 10 and 11, respectively, followed by $n_0 + 3i$ zeros. Let B be the collection of all of these integers, $B = \bigcup_{i=1}^{|A|+2} \{b_i^1, b_i^2, b_i^3\}$.

Our instance of restricted partition is $C = A \cup \{T, S\} \cup B$. Then we have $4(|A| + 2)$ elements, which is even.

Notice that the multiset condition is already satisfied, since the zones are large enough that we never allow carries across zones. If the same element of C is selected twice there will be too many bits in that zone to add up to the desired sum.

Now that we have constructed the instance of the restricted partition problem, we need to show that there is a solution to the 3DM instance if and only if there is a solution to the restricted partition instance. Suppose that there is a solution $M' \subseteq M$ to the instance of 3DM. Then we pick the elements of A corresponding to the elements of M', call this set A'.

Then we pick either the pair b_i^1, b_i^2 or just b_i^3 for each i so that the total number of elements is $|C|/2 = 2(|A| + 2)$. This means we need to pick $|A - A'| + 2$ pairs and $|A'|$ singles, so that the total number of chosen elements is $|A'| + 2|A| - 2|A'| + 4 + |A'| = 2|A| + 4$. Call this set B'. Then one part of the partition $A' \cup \{S\} \cup B'$ and the other part is $(A - A') \cup \{T\} \cup (B - B')$. The sum of both parts is the same.

Suppose that there is a solution to the restricted partition instance. Since the bits in the elements of B are independent of the bits in the elements in $A \cup \{S, T\}$, there must be a subset of A that sums to T. Thus, there is a solution to the 3DM instance. $\square$

We now define a family of graphs to be used as an analogue for integer partitioning. Arrange $2n$ vertices in a circle and label the vertices from 0 to $2n-1$ clockwise starting at the top. Consider the path $0, 1, 2n-1, 2, 2n-2, 3, \ldots, n-1, n$, call this path Z_n. We will show that n copies of Z decompose K_{2n} by repeatedly rotating the path clockwise by 1 vertex (see figure 3).

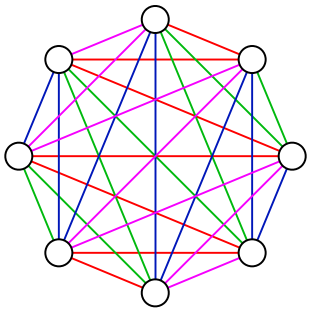


Figure 3: K_8 composed of four copies of Z_4.

We define the graph $Z_{n,k}$ to be k such rotated copies of the path Z_n on $2n$ vertices. We observe the following about these graphs:

Lemma 2. *There exists a partition of the edge set of K_{2n} into the graphs $Z_{n,x_1}, Z_{n,x_2}, \ldots Z_{n,x_c}$ for some positive integers $x_1, x_2, \ldots x_c$, if and only if $x_1 + x_2 + \cdots + x_c = n$.*

Proof. The path Z_n has $2n - 1$ edges and K_{2n} has $2n(2n-1)/2$ edges, so n copies of Z_n gives the exact number of edges needed to decompose K_{2n}.

We want to show that by repeatedly rotating the path clockwise by 1 vertex, n copies of Z_n cover all edges of K_{2n}. Label the vertices 0 to $2n-1$. Then the n rotated copies cover with their first edge the edges $(0, 1), (1, 0), \ldots, (n-1, n)$.

It is sufficient to show that each edge in K_{2n} is an edge of exactly one copy of Z_n. Let (u, v) be an arbitrary edge in K_{2n}. If a rotated copy of Z_n goes through the edge (u, v), it must either hit the edges $(v+1, u), (u, v), (v, u-1)$ or $(v-1, u), (u, v), (v, u+1)$. One of these options will have its first edge directed clockwise and the other will be counterclockwise. Our copies of Z_n always start clockwise. Secondly, there are two possible directions that the path takes, (u, v) or (v, u). One of these paths has its first edge in $(0, 1), \ldots (n-1)$ and the other has its first edge in $(n, n+1), \ldots (2n-1, 0)$. Our paths only have their first edge in $(0, 1), \ldots, (n-1, n)$. $\square$

We are now ready for the main proof.

Theorem 1. *The complete multigraph decomposition problem is NP-complete.*

Proof. The problem is clearly in NP. Given a solution, we can check in polynomial time that each edge of K_n is covered exactly m times.

To show NP-hardnes we reduce from restricted partition. Let S be a set of n positive integers such that n is even and there does not exist a multigraph M of $n/2$ elements of S such that $\sum_{m \in M} m = \frac{1}{2} \sum_{s \in S} s$.

Let $\sigma = \frac{1}{2} \sum_{s \in S} s$, which is the target size of the partition. Then for each positive integer s in S we construct a graph $Z_{\sigma,s}$, and give the multi-edges $n/2$ edges each. We then combine all n of these $Z_{\sigma,s}^{n/2}$ graphs as subgraphs of one graph on $2\sigma n$ vertices. Finally, we add a single edge between each pair of vertices between two different subgraphs. Call this graph H.

We claim that there is an H-decomposition of $\frac{n}{2} K_{2\sigma n}$ if and only if there is a solution to the restricted partition problem.

Suppose that $S' \subset S$ such that $\sum_{s' \in S'} s' = \sigma$. We will use $n/2$ copies of H. We divide our $2\sigma n$ vertices into n subgraphs of 2σ vertices each. For $n/2$ of these subgraphs, we map one of each of the graphs $Z_{\sigma,s'}$ for each s' in S', each graph coming from a different copy.

Likewise, for the other $n/2$ subgraphs, we map one of each of the graphs $Z_{\sigma,s}$ for each s in $S - S'$. Both sets S' and S sum to σ, so by our lemma we can permute the vertices within the subgraphs in such a way that the result on each subgraph is $K_{2\sigma}$ with multiplicity $n/2$. Finally, each of the edges between the subgraphs will be covered by $n/2$ edges, and the overall graph is $\frac{n}{2} K_{2\sigma n}$.

Now suppose that there is a solution to the H-decomposition problem. Then the vertices that make up a subgraph must all be mapped together to another subgraph. To see this, notice that the Z graphs are connected, so if the vertices were split between two Z graphs, there would be $n/2$ edges between the two. But all of these edges are already connected by single edges in the original graph, so the number of edges in each multi-edge would exceed $n/2$. The condition that there does not exist a multiset solution to the partition problem implies that a solution cannot map a subgraph to itself. Thus, there must a set $S' \subset S$ such that the graphs $Z_{\sigma,s'}$ for all $s' \in S'$ form a partition of $K_{2\sigma}$, and so there must be a solution to the partition problem. $\square$

5 Computation

5.1 Algorithm

We present an algorithm that takes a graph G on n vertices and a positive integer m, and determines if there is a G-decomposition of mK_n. We assume that we are given a nontrivial input, that is, the number of edges of G divides the number of edges of mK_n and the gcd of the degrees of G divides the vertex degree of mK_n.

A naive brute-force approach to this problem is to make as many copies of G as are needed and for try all possible permutations of the graph for each copy. This approach runs in $O((n!)^c)$ time. Our algorithm essentially does the same thing, but there are several things we can do to reduce the number of permutations checked. The time complexity for our algorithm is still $O((n!)^c)$.

Let $G_1, G_2, \ldots, G_c$ be the c copies of the graph. We label the vertices 0 through $n-1$. We keep track of the indices of permutations using a $c \times n$ matrix, I.

We can assume that $G_1 = G$, which means that the first permutation is the identity. The order of the permutations does not matter, so we ensure that our c permutations are in lexicographic order.

We fill in values of I, moving left to right, top to bottom. For each row, we keep track of which indices from 0 to $n-1$ have not been used. Then for each value in the matrix we branch, exploring all possible indices. As we do so, we build a multigraph M with the permutated copies of G.

We can terminate a branch if the degree of a vertex ever exceeds $m(n-1)$ or if the multiplicity of an edge exceeds m. Since we are only interested in whether or not a solution exists, if we find a solution we can terminate immediately. The worst case scenario is when no solution exists, and we must explore all possibilities.

The algorithm was implemented in Java, and a github repository with the implementation can be found in Appendix B.

5.2 Results

Table 1 shows the results computed by the algorithm. The algorithm becomes intractable for graphs with 6 or more edges and for graphs on 5 vertices with m greater than 4.

Cameron and Cioabă were interested in graphs like the 3-star, whose partition modulus m_0 (see Section 2) was not equal to the modulus predicted by the divisibility conditions, m_1.

The 3-pan, butterfly, chair, and 4-pan (see Appendix A) all have the expected partition modulus. The cricket has a predicted modulus of 1, but fails $m = 1, 3$. This suggests that the cricket could be another graph where $m_0 \neq m_1$. We would need to show that mK_n cannot be decomposed by the cricked for all odd m. Figure 4 shows the solution for $m = 2$ with the cricket graph.

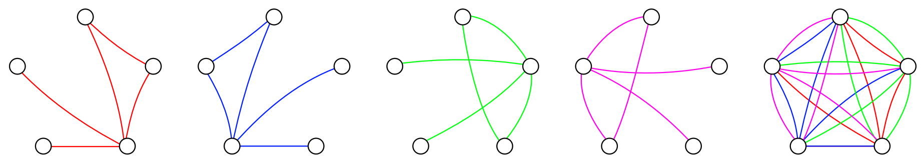


Figure 4: The solution for $m = 2$ with the cricket graph.

In this project, we only tried 5 graphs, but there are more graphs that could feasibly be run on the algorithm on 5 vertices.

6 Conclusion

We would still like to know whether the problem of determining if there is a G-decomposition of mK_n, given a graph G on n vertices and a positive integer m is NP-complete. If this problem is NP-

Graph	n	e	degrees	m_1	$m =$	1	2	3	4	5	6
3-star	4	3	1,1,1,3	1		N†	Y†	N†	Y*	N†	Y*
3-pan	4	4	1,2,2,3	2		-	Y	-	Y*	-	Y*
Butterfly	5	6	2,2,2,2,4	3		-	-	Y	-	-	Y*
Chair	5	4	1,1,1,2,3	2		-	Y	-	Y*	-	Y*
4-pan	5	5	1,2,2,2,3	1		N	Y	Y	Y*	Y*	Y*
Cricket	5	5	1,1,2,2,3	1		N	Y	N	Y*	?	Y*

Table 1: Results for a graph G on n vertices with e edges. We give the degrees of the vertices of G, and m_1, the smallest integer for which the divisibility conditions are satisfied. Y denotes that the problem is solvable, N denotes that it is not. - denotes that the problem does not satsify the divisibility conditions and is not solvable. * denotes that the result is implied. † denotes results shown by Cameron and Cioabă. ? denotes values not computed.

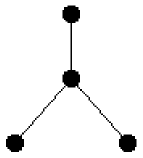
complete, then the NP-completeness of the problem where G is a multigraph would follow since every graph is a multigraph.

There are ways to improve the algorithm. We could exploit the symmetries in the given graph G, since if it has symmetries, there are certain permutations that result in the same graph. We could also introduce heuristics for how we explore the tree of possible solutions, which would not improve the worst case time, but could make it faster to find solutions if a solution exists.

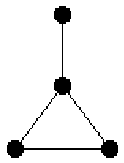
Our algorithm is exact, meaning that it always returns the correct solution. For practical purposes, it is not always necessary to have a correct solution, and there can be polynomial time approximate solutions to NP-complete problems. We could phrase the problem as a minimization problem, where we take isomorphic copies of a graph and want to minimize the number of edges of the resulting graph that go over or under the target value of m. Graph decomposition of mK_n has applications in experimental design, so it could be potentially useful to design an algorithm tailored for this use case.

A Graphs

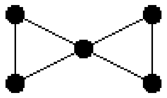
- 3-star



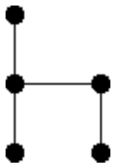
- 3-pan



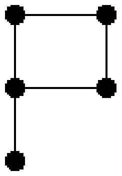
- Butterfly



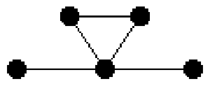
- Chair



- 4-pan



- Cricket



B Code implementation

An implementation of the described algorithm can be found at `https://github.com/BenjaminGustafson/graph_partition`.

References

[CC15] Sebastian M Cioabă and Peter J Cameron. A graph partition problem. *The American Mathematical Monthly*, 122(10):972–982, 2015.

[Coo71] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[DT97] Dorit Dor and Michael Tarsi. Graph decomposition is np-complete: A complete proof of holyer's conjecture. *SIAM Journal on Computing*, 26(4):1166–1187, 1997.

[GJ79] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[Hol81] Ian Holyer. The np-completeness of some edge-partition problems. *SIAM Journal on Computing*, 10(4):713–717, 1981.

[Wil75] Richard M. Wilson. Decomposition of a complete graph into subgraphs isomorphic to a given graph. *Proceedings of the Fifth British Combinatorial Conference, University of Aberdeen, Aberdeen*, pages 647–659, 1975.