On Finding Hajós Constructions

by

Erik Johnson

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

We develop a method for searching for Hajós constructions. Our results include the discovery of new constructions for some well-known graphs, including the Grötzsch graph, Chvátal graph, and Brinkmann graph; also, we prove that the first two of these are shortest possible constructions. These are the first published constructions for these graphs.

Acknowledgments

Thank you Dr. Joseph Culberson for getting me started on this topic and for your ongoing help and interest. Thank you Dr. Ryan Hayward for seeing this project through and for all of your help in advising me on my thesis.

I would also like to thank Jeanette Bliemel for her patience, support, and encouragement as I took my meandering path through work and academics.

Table of Contents

1 Introduction **1**
- 1.1 Graph Colouring and Complexity Theory . . . 2
- 1.2 Contributions . . . 7

2 The Grötzsch Graph **8**
- 2.1 Background . . . 8
- 2.2 Construction . . . 9
- 2.3 Minimality of the Construction . . . 9
- 2.4 Alternate Constructions of the Grötzsch Graph . . . 16

3 Variations on Hajós Constructions **18**
- 3.1 Tree-Like Hajós Constructions . . . 18
- 3.2 Hajós' Proof of Completeness . . . 22
- 3.3 Top-Down Constructions . . . 23

4 Searching for Hajós Constructions **27**
- 4.1 Background . . . 27
- 4.2 Hajós' Algorithm . . . 28
- 4.3 Searching the Constructions Space . . . 29
- 4.4 Isomorphisms and Tree-Like Constructions . . . 31
- 4.5 A Practical Approach . . . 35

5 Example Constructions **40**
- 5.1 The Chvátal Graph . . . 40
- 5.2 The 13-Cyclotomic Graph . . . 42
- 5.3 The Brinkmann Graph . . . 44
- 5.4 The 5-Chromatic Mycielski Graph . . . 48

6 Hajós and Ore **52**
- 6.1 Hajós Versus Ore . . . 56
- 6.2 Top-Down Search with Vertex Merges . . . 57

7 A Sequence of (k-1)-Clique-Free k-Chromatic Graphs **60**

8 Conclusions and Future Work **64**
- 8.1 Conclusions . . . 64
- 8.2 Future Work . . . 65

Bibliography **66**

List of Tables

2.1 Vertex merge options for generating triangle-free graphs from the graph of Figure 2.6. 15

5.1 Operations of a Hajós construction of the Brinkmann graph. . 45
5.2 Operations of a Hajós construction of M_5. 50
5.3 Operations to Hajós construct M_5, continued. 51

List of Figures

1.1 A Hajós sum. . . . 3
1.2 A Hajós construction of the 7-wheel, showing the details of the Hajós sum and vertex identifications of the construction. . . . 5
1.3 The 7-wheel construction in Ore merge operations. . . . 5

2.1 The Grötzsch graph. . . . 8
2.2 A shortest Hajós construction of the Grötzsch graph. . . . 10
2.3 Graphs generated by Hajós summing 4-cliques. . . . 11
2.4 Candidate graphs for constructing graphs in 2 Hajós sums. . . . 12
2.5 Constructing graphs where all triangles share a single edge. . . . 13
2.6 Triangle-free graphs generated from the graphs in Figure 2.5. . . . 14
2.7 Another length-four construction of the Grötzsch graph. . . . 17

3.1 A regular Hajós construction, tree-like Hajós construction, DAG map, and tree map. . . . 19
3.2 DAG and tree maps of our construction for the Grötzsch graph. 20
3.3 Two DAG reductions of the same tree-like construction. . . . 21
3.4 The add-edge operation. . . . 24
3.5 Reversing the add-edge operation. . . . 24
3.6 Finding a 7-wheel Hajós construction using add-edge. . . . 25
3.7 Reducing a construction to required subgraphs. . . . 26

4.1 An illustration of the difficulty in optimizing a top-down search. 32

5.1 The Chvátal graph. . . . 40
5.2 A shortest construction of the Chvátal graph. . . . 41
5.3 The DAG map for Figure 5.2. . . . 42
5.4 The 13-Cyclotomic graph. . . . 42
5.5 A DAG map for the 13-Cyclotomic graph. . . . 42
5.6 A shortest construction of the 13-Cyclotomic graph. . . . 43
5.7 The Brinkmann graph. . . . 44
5.8 A DAG map for the Brinkmann graph, found by our algorithm. 44
5.9 Graphs of our construction of the Brinkmann graph. . . . 46
5.10 Graphs of our construction of the Brinkmann graph, continued. 47
5.11 M_5: the 5-chromatic Mycielski graph. . . . 48
5.12 DAG map of a length-24 construction of M_5. . . . 49

6.1 Simulating an Ore merge in HC-. . . . 56
6.2 A demonstration of a smaller Hajós construction and a larger Ore construction for a graph. . . . 58

7.1 The 5-chromatic, 6-chromatic, and 7-chromatic graphs of our sequence. . . . 60
7.2 Our construction of C_5. . . . 62

Listings

4.1 Hajós' algorithim. 29
4.2 Search for a minimum tree-like supergraph construction. . . . 30
4.3 Greedily reduce a tree-like construction to a DAG and return its length. 31
4.4 Find all possible top-down constructions. 33
4.5 Find a minimum length Ore construction. 34
4.6 Find a non-branching construction. 36
4.7 Depth limited leaf search. 37
4.8 Find a construction with branches of specified depth limit. . . 39

Chapter 1

Introduction

In 1981, Mansfield and Welsh investigated the graph construction system of Hungarian mathematician György Hajós and found interesting complexity theory implications. They noted that a Hajós construction of a graph can be used to provide a proof of a lower bound on the graph's chromatic number [14]. A sequence of Hajós construction steps that begins with a k-clique and ends in a graph G provides a certificate that G cannot be k-1 coloured. This seems surprising, as Graph Colouring is NP-complete, and it is not believed to be possible in general to provide polynomially verifiable certificates of negative results to NP-complete problems. If polynomially sized Hajós constructions exist for all graphs, it would imply that NP = CoNP, which is considered unlikely.

As part of their investigation, Mansfield and Welsh noted that outside of a few special cases it seems hard to find a construction for a given arbitrary graph, describing the problem as being "exceptionally difficult" [14]. Based on this challenge, they defined the Hajós number of a graph as the length of the minimum Hajós construction for the graph.

In the literature we find few examples of nontrivial Hajós constructions and little further work on finding Hajós constructions or Hajós numbers.

We are interested in problems related to finding Hajós numbers and short Hajós constructions for graphs. We investigate algorithms for generating constructions of graphs and the challenges involved in combinatorial search approaches to finding short constructions. We also investigate finding minimum

Hajós constructions and proving the value of the Hajós number of a graph. We look at some short constructions for various graphs that we have found, including a minimum construction and Hajós number for the Grötzsch graph, answering a question posed by Mansfield and Welsh.

1.1 Graph Colouring and Complexity Theory

A vertex k-colouring of a graph can be considered a proof that the graph is k-colourable; moreover, this proof of k-colorability can be quickly—i.e. in time polynomial in the size of the graph—verified: it suffices to confirm, for each pair of adjacent vertices, that the colours assigned to the vertices differ. By contrast, for graphs that are not k-colourable, there is no known proof that confirms non-k-colorability and that can be verified in polynomial time.

For those familiar with complexity theory, the previous paragraph can be summarized: k-colorability is in the class NP and is not known to be in the class co-NP. See Garey and Johnson for a discussion of the class NP [7].

As Mansfield and Welsh observe, every k-chromatic graph has a Hajós construction from k-cliques, and such a construction is a proof that the graph is not (k-1)-colourable. Verifying that a k-construction is valid is a proof that the graph is not (k-1)-colourable, and the time to check the proof is polynomial in the size of the construction.

If every k-chromatic graph has a Hajós construction whose size is polynomial in the size of the graph, then every graph can have its k-chromaticity confirmed in polynomial time. This would imply that NP=co-NP, which is considered unlikely. This raises the question: how does one find a short Hajós construction?

Pitassi and Urquhart characterize the Hajós construction in terms of a proof system known as the Hajós calculus. They show that the Hajós calculus is equivalent in power to extended Frege proof systems, a powerful class of proof systems for the propositional calculus. Specifically, they show that "the Hajós calculus is polynomially-bounded if and only if extended Frege proof systems are polynomially bounded" [18]. This indicates that the complexity

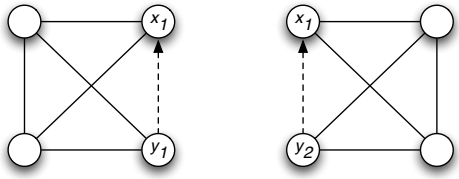
questions of the Hajós calculus are likely difficult to resolve.

A *graph* G is an ordered pair of disjoint sets (V, E) such that E is a set of unordered pairs of V. The set V is the set of *vertices* and the set E is the set of *edges* [2]. If the edge (x, y) exists, then vertices x and y are said to be *adjacent*. $G = (V, E)$ is *isomorphic* to $G' = (V', E')$ if there is a bijection $\gamma : V \leftrightarrow V'$ such that $(x, y) \in E$ iff $(\gamma(x), \gamma(y)) \in E'$ [2].

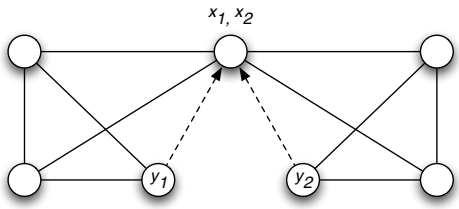
For graph theory terms that we do not define here, refer to Bollobás [2].

Definition 1.1 (Hajós sum)**.** Given graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with disjoint vertex sets, an edge $(x_1, y_1) \in E_1$, and an edge $(x_2, y_2) \in E_2$, the *Hajós sum* $G_3 = G_1(x_1, y_1) +_H G_2(x_2, y_2)$ is the graph obtained as follows: begin with $G'_3 = (V_1 \cup V_2, E_1 \cup E_3)$; then in G'_3 delete edges (x_1, y_1) and (x_2, y_2), identify vertices x_1 and x_2, and add edge (y_1, y_2). Now define G_3 as the current G'_3.

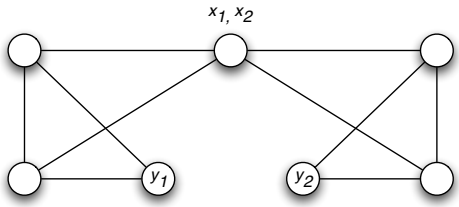
See Figure 1.1, where a Hajós sum is performed on two 4-cliques, producing the *Hajós graph*.



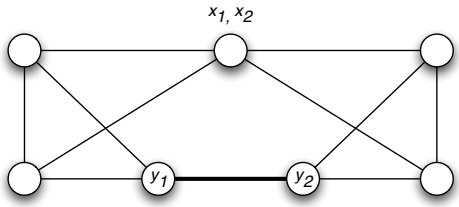
(a) Take two disjoint graphs. The dotted arrows denote the edges that we will perform the sum on.



(b) Identify vertices x_1, x_2.



(c) Delete edges x_1, y_1 and x_2, y_2.



(d) Add the edge y_1, y_2, denoted with a bold line. The unlabelled version of the resulting graph is known as the Hajos graph.

Figure 1.1: A Hajós sum.

We now define an operation due to Ore [17, 19] that is useful because it expresses Hajós constructions in a more compact form.

Definition 1.2 (Ore merge)**.** Given graphs G_1 and G_2 with disjoint vertex sets, an edge (x_1, y_1) of G_1, an edge (x_2, y_2) of G_2, a vertex subset S_1 of $G_1 - x_1$, and a bijection $\gamma : S_2 \leftrightarrow S_1$ where S_2 is a subset of $G_2 - x_2$, and $<y_2, y_1>$ does not exist in γ, the *Ore merge* $G_3 = O_+(G_1, G_2, (x_1, y_1), (x_2, y_2), S_1, S_2, \gamma)$ is the graph obtained by the Hajós sum $G_3 = G_1(x_1, y_1) +_H G_2(x_2, y_2)$ followed by identifying vertices $v, \gamma(v)$ for all v in S_2.

Definition 1.3 (Hajós construction)**.** Given a graph G and an integer k, a *k-Hajós-construction* of G is a sequence of graphs $G_0, \ldots, G_t$, where G_0 is a k-clique, $G_t = G$, and for each $j \geq 1$, one of the following holds:

- for some $x <= y < j$, G_j an Ore merge of G_x and G_y, or
- for some $x < j$, G_j is G_x with two non-adjacent vertices identified, or
- for some $x < j$, G_j is a supergraph of G_x.

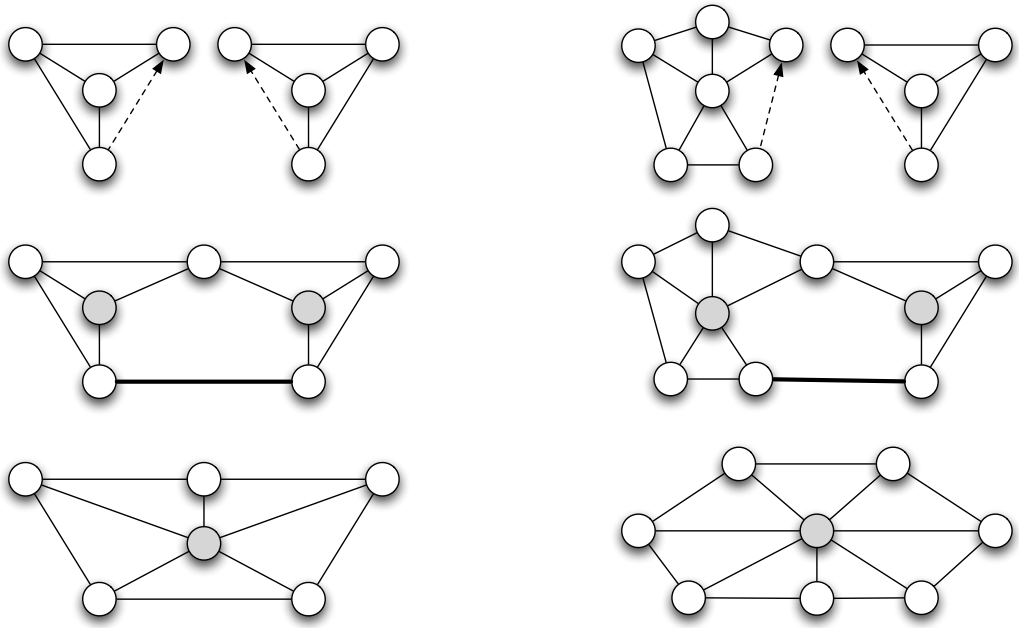
See Figure 1.2 for an example of a Hajós construction. In our figures, we use a dashed arrow to indicate the edges that are used as inputs to a Hajós sum, where the heads of the arrows are the vertices that will be merged.

Definition 1.4 (Ore construction)**.** Given a graph G and an integer k, a *k-Ore-construction* of G is a sequence of graphs $G_0, \ldots, G_t$, where G_0 is a k-clique, $G_t = G$, and for each $j \geq 1$,

- either, for some $x <= y < j$, G_j is an Ore merge of G_x and G_y, or
- for some $x < j$, G_j is a supergraph of G_x.

We also make use of the following terms.

Chromatic number $\chi(G)$ is the smallest number of colours, k, that can be used to colour a graph, G, such that for each pair of adjacent vertices in G, the colours assigned to the vertices differ. A graph is *k-chromatic* if $\chi(G) = k$.



(a) Begin with two copies of K_4. Hajós sum the graphs and identify the interior vertices, marked in grey, to produce the 5-wheel.

(b) Hajós sum the 5-wheel and another K_4, and again identify the interior vertices in the resulting graph. This produces the 7-wheel.

Figure 1.2: A Hajós construction of the 7-wheel, showing the details of the Hajós sum and vertex identifications of the construction.

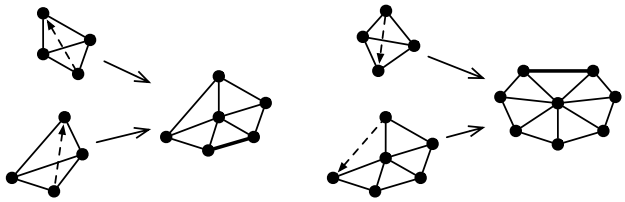


Figure 1.3: The 7-wheel construction in Ore merge operations.

Criticality A k-chromatic graph G is *k-critical* if removing any edge from G results in a graph that is not k-chromatic.

Girth The *girth* of a graph is the length of the shortest cycle in the graph. For example, a graph that contains a cycle of length four, but no triangles, has a girth of four.

Constructability A graph G is *Hajós k-constructible* if there is a Hajós construction that begins with the k-clique and produces G.

Construction Length The *length* of a Hajós or Ore construction is the number of applications of the operations listed in the definition of the respective construction [18].

Monotonicity A construction is *monotonically increasing* if any graph generated by an operation in the construction has at least as many vertices as any of its inputs.

Hajós Number $h_k(G)$, is the length of the minimum Hajós k-construction of a k-chromatic graph G [14].

For example, it follows that the 7-wheel construction of Figure 1.3 has a construction length of two.

Following the approach of Mansfield and Welsh, we define Hajós constructions in terms of the Ore merge. In a Hajós construction, the Ore merge operation can identify any number of vertices so long as, for each pair of vertices being merged, the vertices are not from the same input graph. Conversely, the vertex identification operation of a Hajós construction—which does not exist as an independent operation in the Ore construction—allows any non-adjacent vertices to be merged. This distinction has implications for the complexity of the Hajós construction, as we will see in Chapter 6. Ore constructions use the same Ore merge and supergraph operations as Hajós constructions, but they do not allow the independent vertex identification operation. Thus, all Ore constructions are Hajós constructions, but not all Hajós constructions are Ore constructions.

Notice that an Ore merge consists of a Hajós sum followed by a vertex identification step. Also notice that a Hajós sum that is not followed by a vertex identification step is equivalent to an Ore merge with an empty set of vertices identified. Often, we will discuss Ore merges in terms of their Hajós sum and vertex identification parts, as shown in Figure 1.3. When we discuss the length of a construction the Hajós sum and vertex identification step are counted as one Ore merge operation.

Mansfield and Welsh pose finding the Hajós number of a graph as a problem with unknown complexity. This problem is known to be at least NP-hard, but

it is possible that it is not in P-SPACE if the Hajós calculus is not polynomially bounded [14].

1.2 Contributions

Mansfield and Welsh use the Grötzsch graph to demonstrate the difficulty of finding the Hajós number of a graph and leave finding a construction as an open problem. In Chapter 2, we give a length-four construction of the Grötzsch graph, and we prove that this length is the minimum, thus proving the Hajós number of the graph is four.

Based on Hajós' original proof of completeness, we describe a top-down approach to finding constructions for graphs in Chapter 3. Building on these ideas, we develop algorithms for computer search for short and minimum Hajós constructions in Chapter 4.

In Chapter 5, we apply our search approach to some well-known graphs. Our results include a short construction for the Brinkmann graph and a minimum construction of the Chvátal graph. The code used to find these constructions is available at `https://github.com/eej/hajos-search`.

In investigating the Ore construction, we find an equivalence with a construction defined by Pitassi and Urquhart. This equivalence proves that the Ore calculus is not polynomially bounded and demonstrates a complexity difference between the Hajós and Ore constructions. In Chapter 6, we prove this equivalence and look at the implied differences between the Hajós and Ore constructions.

In Chapter 7, we discuss a sequence of short k-chromatic, (k-1)-clique-free graphs that can be generated via a simple Hajós construction.

Chapter 2

The Grötzsch Graph

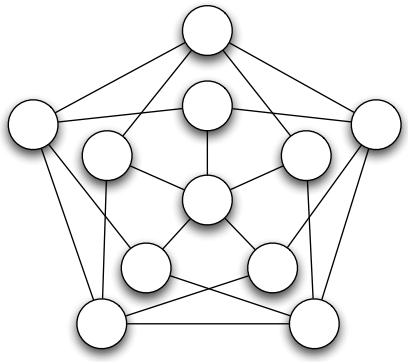


Figure 2.1: The Grötzsch graph.

2.1 Background

The Grötzsch graph, shown in Figure 2.1, is the smallest triangle-free 4-chromatic graph [8, 5]. It is also the only triangle-free, 4-chromatic graph with eleven vertices.

In addition to Mansfield and Welsh's use in illustrating the difficulty of finding Hajós numbers, Liu and Zang use the Grötzsch graph to generate graphs that are difficult to colour using common search approaches [13]. Liu and Zang generate large triangle-free 4-chromatic graphs using Hajós sums with the Grötzsch graph as a base, but they do not provide full Hajós constructions from cliques for their graphs.

2.2 Construction

In Figure 2.2, we show a length four Hajós construction for the Grötzsch graph. In this figure our graphs have labeled vertices. Hajós constructions are generally considered in the context of unlabelled graphs; however, for illustrative purposes, we show many example graphs with labels, and we allow vertex relabelling. Additionally, for clarity we illustrate the Hajós sum and vertex identification operations as separate steps. However, as noted in Section 1.1, these sum and identification steps represent the component steps of a single Ore merge operation.

2.3 Minimality of the Construction

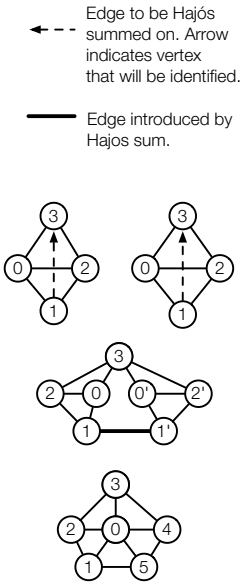
The construction in Figure 2.2 is minimum, and the Hajós number of the Grötzsch graph is four. To prove this, we show that the Grötzsch graph cannot be constructed in one, two, or three steps. It is trivial to show that one step is not possible: a Hajós sum of two 4-cliques yields a graph with seven vertices, while the Grötzsch graph has 11.

To show that it is not possible to construct the graph in two or three steps we rely on the following property.

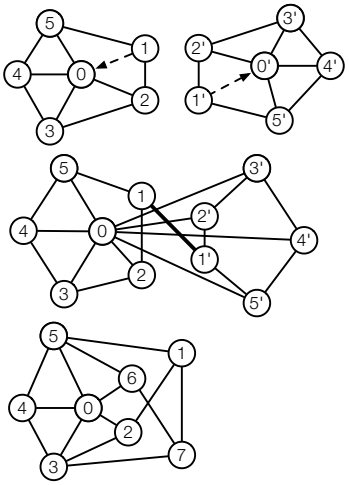
Lemma 2.3.1. *Any Hajós construction of a triangle-free graph must include at least one graph G such that G has an edge (a, b) in which all triangle subgraphs include (a, b).*

Proof. Consider how triangle-free graphs are formed in Hajós constructions. All constructions begin with cliques which have multiple triangle subgraphs. Thus, at some point in the construction an operation must take a graph that contains triangles and produce a graph that is triangle-free. We examine how this transition to triangle-free graphs can occur.

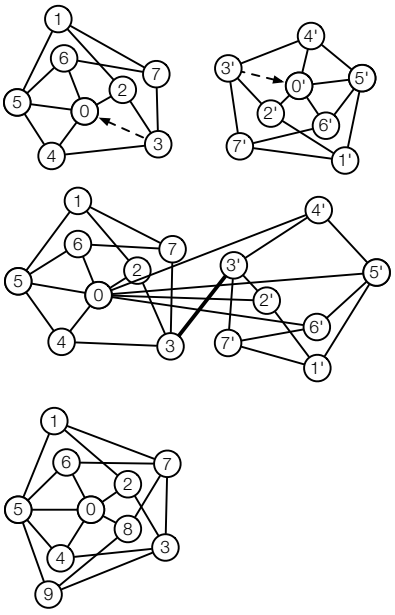
First, consider the vertex identification operation. Adjacent vertices cannot be merged, so a triangle subgraph cannot be eliminated by collapsing the triangle through vertex identification. Also, merging other vertices in the graph cannot eliminate a triangle subgraph. Multiple triangle subgraphs can



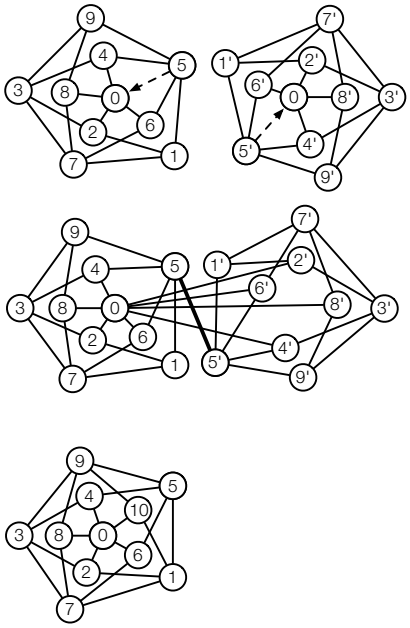
(a) Hajós sum two 4-cliques to form the Hajós graph. Identify $0' \to 0$ to create the 5-wheel. Relabel: (1', 2') $\mapsto$ (5, 4).



(b) Take two copies of the 5-wheel. Hajós sum $(0, 1)$ with $(0', 1')$. Identify: (3', 4', 5') $\to$ (5, 4, 3). Relabel: (1', 2') $\mapsto$ (7, 6).



(c) Take two copies of the graph constructed in 2.2b. Hajós sum $(0, 3)$ with $(0', 3')$. Identify: (3', 4', 5', 6', 7') $\to$ (7, 6, 5, 4, 3). Relabel: (1', 2') $\mapsto$ (9, 8).



(d) With two copies from 2.2c, Hajós sum $(0, 5)$ with $(0', 5')$. Identify: (1', 2', 3', 4', 5', 7', 8', 9') $\to$ (5, 4, 3, 2, 1, 9, 8, 7). Relabel: $6' \mapsto 10$.

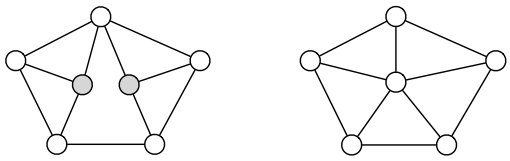
Figure 2.2: A shortest Hajós construction of the Grötzsch graph, found by hand. This is construction has length four.

be merged into a single triangle, but at least one triangle will remain, and it cannot be eliminated via vertex merges. Next, consider the supergraph operation. If a graph G_1 contains a triangle, any supergraph of G_1 also contains this triangle, and this operation cannot produce new triangle-free graphs in a Hajós construction.

Thus, a Hajós sum is the only operation that can eliminate triangles. Consider the sum of a graph G_1 with a triangle $\{a, b, c\}$ and a graph G_2. If the sum is on an edge in $\{a, b, c\}$, that edge is removed in the resulting graph G and $\{a, b, c\}$ will not induce a triangle in G. Conversely, we can see that in the case of a sum operation on an edge not in $\{a, b, c\}$, then $\{a, b, c\}$ still induces a triangle in G. This argument holds for all triangles in G_1. Thus, for all triangle subgraphs of G_1 to be removed in G, all triangles in G_1 must share a common edge on which a Hajós sum will be performed. By symmetry, this argument also applies to G_2, where any triangles in G_2 must share a common edge that will be eliminated by a Hajós sum operation for G to be triangle-free.

Thus, to generate any triangle-free graph in a Hajós construction, we must first generate at least one graph where all triangle subgraphs share a single edge.

□



(a) Hajós graph (b) 5-wheel

Figure 2.3: Graphs generated by Hajós summing 4-cliques.

Using Lemma 2.3.1, we evaluate the triangle-free graphs that are obtainable in three operations and show the following.

Lemma 2.3.2. *Every 4-chromatic triangle-free graph with Hajós number less than four has 14 or more vertices.*

Proof. A Hajós sum on two 4-cliques gives the Hajós graph. See Figure 2.3a. Identifying the two vertices marked in grey in this figure produces the 5-wheel.

See Figure 2.3b. Any vertex identification on the Hajós graph produces either the 5-wheel or a supergraph of the 4-clique. Any identification on the 5-wheel produces a supergraph of the 4-clique. None of the graphs obtainable from a single Hajós sum have all triangles sharing a single edge. Thus, a triangle-free graph cannot be produced from two Hajós sums. Further, we have defined the set of candidate graphs that can be used to produce graphs in three Hajós sums. See Figure 2.4.

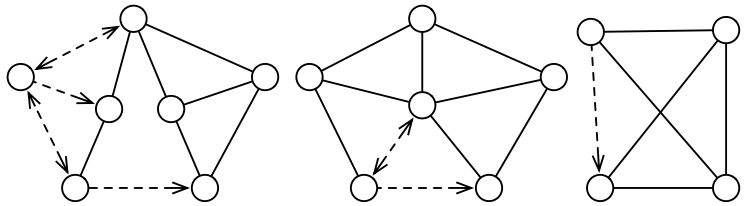
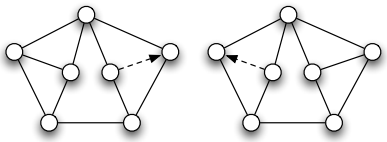


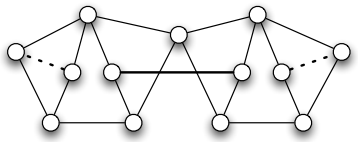
Figure 2.4: Candidate graphs for constructing graphs in 2 Hajós sums. Dashed arrows indicate edges that we consider for Hajós sum inputs. Any other sum operations on edges in these graphs yield graphs that are isomorphic to graphs yielded by sums on these edges.

We consider triangle-free graphs that can be generated from three Hajós sum operations. As an intermediate step, we consider graphs obtainable from the graphs in Figure 2.4, where all triangles share a single edge. In the 5-wheel and 4-clique graphs, there are multiple independent triangle subgraphs, and eliminating a single edge leaves multiple triangles. However, the Hajós graph has four triangles, and the triangles form two pairs such that each pair of triangles share an edge. In Figure 2.5, we show how this can be used to construct a graph where all triangles share a single edge.

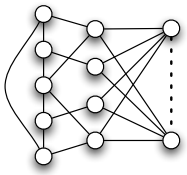
The Hajós sum performed in Figure 2.5a yields a graph where all triangle subgraphs share one of two edges, and the triangle subgraphs form two disjoint sets. See Figure 2.5b. In this graph the closest vertices between the two sets of triangles have an intermediate vertex between them. This allows the two sets of triangles to be merged into one without introducing new independent triangles. No other Hajós sum operations performed on our candidates from Figure 2.4 have this property. Thus, the graphs in Figures 2.5c and 2.5d are the only graphs obtainable in two Hajós sum operations where all triangle subgraphs share a single edge.



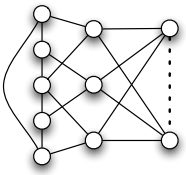
(a) Begin by Hajós summing two Hajós graphs. The dashed arrows represent the edges to be summed on.



(b) The resulting graph has the introduced edge shown in bold. Every triangle subgraph shares one of the two edges that are shown as dotted lines. We merge the vertices of these edges.



(c) In the resulting graph, every triangle subgraph shares the edge shown as a dotted line.



(d) The two centre vertices from the previous graph can be identified, resulting in an additional graph where all triangles share a single edge. Any other vertex identifications introduces independent triangles.

Figure 2.5: Constructing graphs where all triangles share a single edge starting from two copies of the Hajós graph.

Figure 2.6 shows the triangle-free graphs obtainable from the graphs in Figure 2.5. The two output graphs in Figure 2.5 are the only graphs obtainable in two Hajós sums that satisfy Lemma 2.3.1. Thus, the graphs in Figure 2.6, in addition to any triangle-free graphs obtainable through vertex identifications of these graphs, form the complete set of triangle-free graphs obtainable in three Hajós sums.

In Figure 2.6, we label the largest of the graphs to discuss vertex merges. Similar to how the graph of Figure 2.5d is constructed via a vertex merge on the graph of Figure 2.5c, the two smaller graphs in Figure 2.6 can be obtained by merging vertices 7 and 8, vertices 14 and 15, or both pairs of vertices. Thus, any triangle-free graph that can be constructed in three Hajós sums can be obtained through vertex merges on the bottom graph in Figure 2.6.

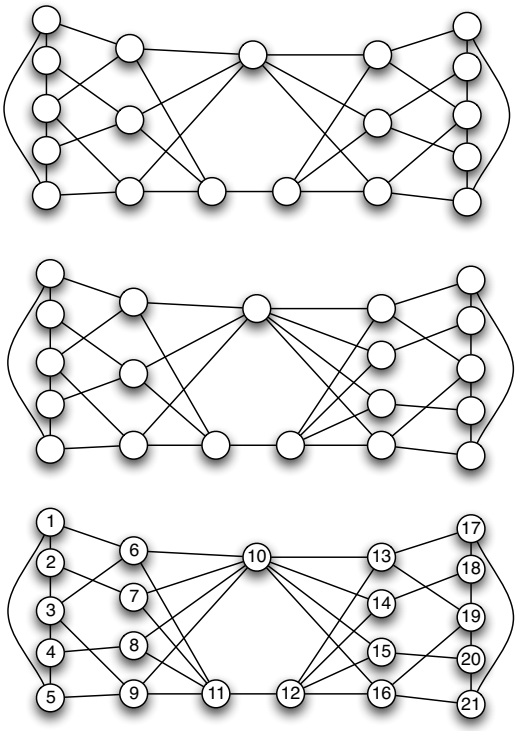
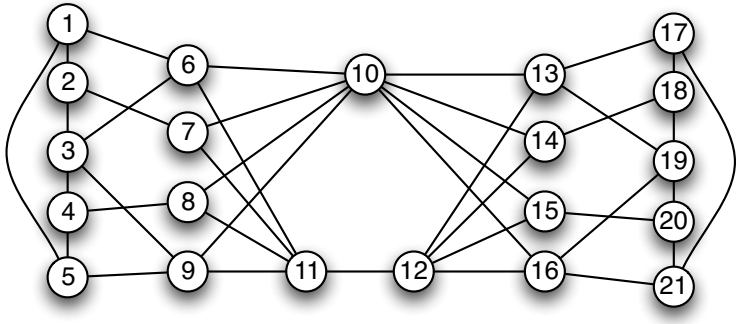


Figure 2.6: Triangle-free graphs generated from the graphs in Figure 2.5.

We now consider graphs that can be obtained through vertex merges on our generated graph. Table 2.1 lists all possible vertex merges and shows whether the resulting graph has a triangle.

As already noted, vertices 7 and 8 can be merged as well as 14 and 15. Also, vertices in the 5-cycle (1, 2, 3, 4, 5) can be merged with the vertices in the 5-cycle (17, 18, 19, 20, 21). An important restriction on this second set of merges is that each vertex from the first 5-cycle can only merge with one vertex from the second 5-cycle. After the two 5-cycles are merged, no further merges can result in triangle-free graphs. Thus, we can reduce the vertex count by five vertices by merging the two 5-cycles, and we can further reduce the count by two through merging 7, 8, and 14, 15. This results in a 14-vertex graph, the smallest triangle-free graph that one can generate through vertex merges. Thus, the smallest triangle-free graph obtainable from three Hajós sums has 14 vertices. □



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	X	—	1, 3, 4	1, 2, 3	—	—	1, 6, 9	1, 4, 5	1, 2, 3	1, 2, 7	1, 2, 7	1, 6, 11	1, 6, 10	1, 6, 10	1, 6, 10	1, 6, 10	✓	✓	✓	✓	✓
2		X	—	1, 2, 4	2, 3, 4	2, 6, 9	—	2, 3, 4	1, 2, 5	1, 2, 6	1, 2, 6	2, 7, 11	2, 7, 10	2, 7, 10	2, 7, 10	2, 7, 10	✓	✓	✓	✓	✓
3			X	—	1, 2, 3	—	3, 6, 9	3, 6, 9	—	2, 3, 7	2, 3, 7	3, 6, 11	3, 6, 10	3, 6, 10	3, 6, 10	3, 6, 10	✓	✓	✓	✓	✓
4				X	—	1, 4, 5	2, 3, 4	—	4, 8, 9	3, 4, 6	3, 4, 6	4, 8, 11	4, 8, 10	4, 8, 10	4, 8, 10	4, 8, 10	✓	✓	✓	✓	✓
5					X	3, 4, 5	1, 2, 5	5, 8, 9	—	1, 5, 6	1, 5, 6	5, 9, 11	5, 9, 10	5, 9, 10	5, 9, 10	5, 9, 10	✓	✓	✓	✓	✓
6						X	1, 2, 6	3, 4, 6	1, 5, 6	—	—	6, 10, 12	6, 11, 12	6, 11, 12	6, 11, 12	6, 11, 12	6, 10, 13	6, 10, 14	6, 10, 13	6, 10, 15	6, 10, 16
7							X	✓	2, 3, 7	—	—	7, 10, 12	7, 11, 12	7, 11, 12	7, 11, 12	7, 11, 12	7, 10, 13	7, 10, 14	7, 10, 13	7, 10, 15	7, 10, 16
8								X	3, 4, 8	—	—	8, 10, 12	8, 11, 12	8, 11, 12	8, 11, 12	8, 11, 12	8, 10, 13	8, 10, 14	8, 10, 13	8, 10, 15	8, 10, 16
9									X	—	—	9, 10, 12	9, 11, 12	9, 11, 12	9, 11, 12	9, 11, 12	9, 10, 13	9, 10, 14	9, 10, 13	9, 10, 15	9, 10, 16
10										X	10, 11, 12	6, 10, 11	—	—	—	—	10, 14, 17	10, 13, 17	10, 14, 18	10, 13, 19	10, 13, 17
11											X	—	6, 10, 11	6, 10, 11	6, 10, 11	6, 10, 11	11, 12, 13	11, 12, 14	11, 12, 13	11, 12, 15	11, 12, 16
12												X	—	—	—	—	12, 14, 17	12, 13, 17	12, 14, 18	12, 13, 19	12, 13, 17
13													X	13, 16, 17	13, 18, 19	13, 16, 20	—	10, 13, 14	—	10, 13, 15	10, 13, 16
14														X	✓	14, 17, 18	10, 13, 14	—	10, 13, 14	10, 14, 15	10, 14, 16
15															X	15, 18, 19	10, 13, 15	10, 14, 15	10, 13, 15	—	10, 15, 16
16																X	10, 13, 16	10, 14, 16	—	10, 15, 16	—
17																	X	—	16, 17, 20	13, 17, 19	—
18																		X	—	17, 18, 20	16, 18, 19
19																			X	—	13, 17, 19
20																				X	—
21																					X

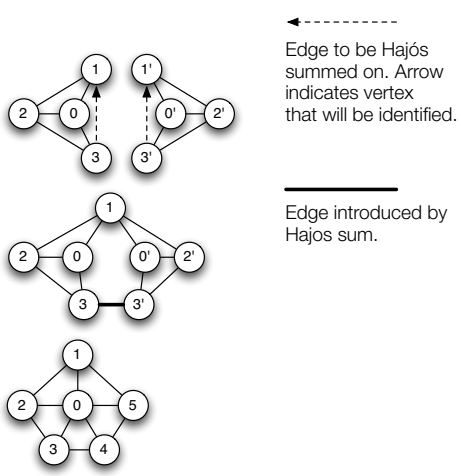
Table 2.1: Vertex merge options. ✓ indicates a vertex pair that can be merged without creating a triangle. A list indicates a triangle that would be created.

Theorem 2.3.3. *The Grötzsch graph has Hajós number four.*

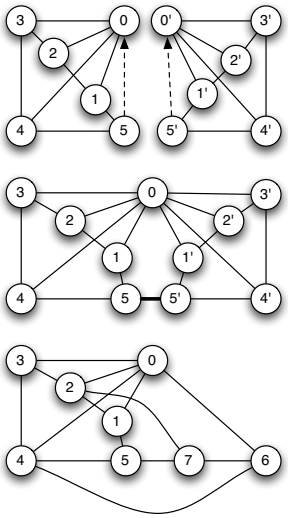
Proof. Since the Grötzsch graph has 11 vertices, by Lemma 2.3.2 we conclude that the Grötzsch graph cannot be obtained in three or fewer Hajós operations. We have demonstrated a length four construction; thus, the Hajós number of the Grötzsch graph is four. □

2.4 Alternate Constructions of the Grötzsch Graph

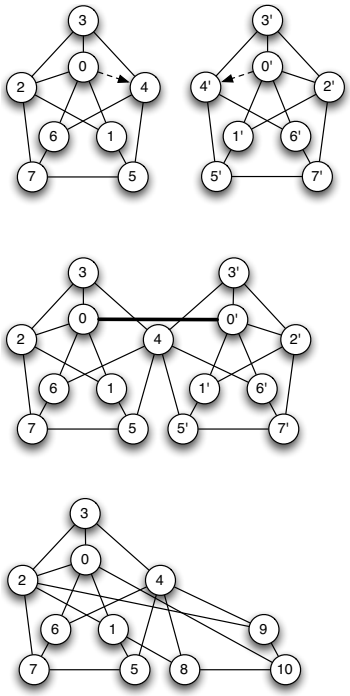
The construction shown in Figure 2.2 is minimum but not unique. Other length-four constructions exist. See Figure 2.7. There the construction begins by generating the 5-wheel graph from 4-cliques; however, the intermediate graphs differ from those of Figure 2.2.



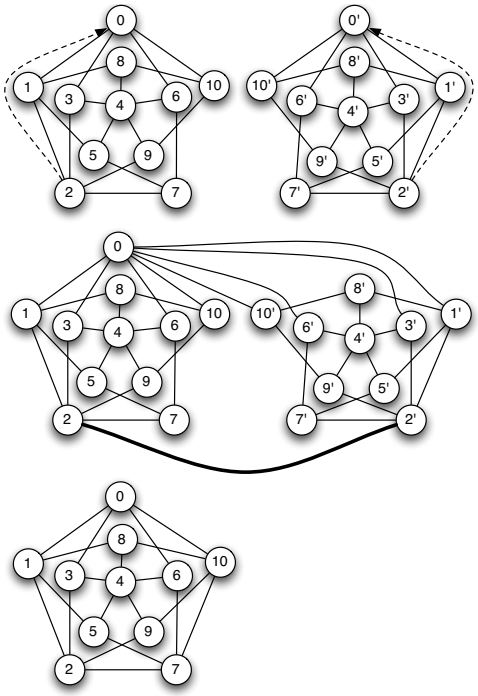
(a) Hajós sum two 4-cliques to form the Hajós graph. Identify 0' → 0 to create the 5-wheel. Relabel: (1', 2') ↦ (4, 5).



(b) Take two copies of the 5-wheel. Hajós sum $(0, 5)$ with $(0', 5')$. Identify: (1', 2', 3') → (2, 3, 4). Relabel: (4', 5') ↦ (6, 7)



(c) Take two copies of the graph constructed in 2.7b. Hajós sum $(4, 0)$ with $(4', 0')$. Identify: (0', 1', 3', 2') → (2, 1, 3, 0). Relabel: (5', 6', 7') ↦ (8, 9, 10).



(d) With two copies from 2.7c, Hajós sum $(0, 2)$ with $(0', 2')$. Identify: (1', 2' , 3', 4', 5', 6', 7', 8', 9', 10') → (10, 7, 6, 4, 9, 3, 2, 8, 5, 1).

Figure 2.7: Another length four construction of the Grötzsch graph found by the search techniques discussed in Chapter 4.

Chapter 3

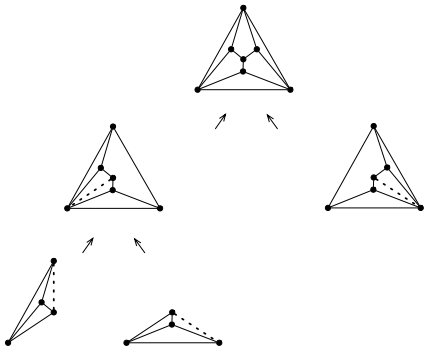
Variations on Hajós Constructions

Before investigating the process of finding Hajós constructions for graphs, we look at different forms constructions can take.

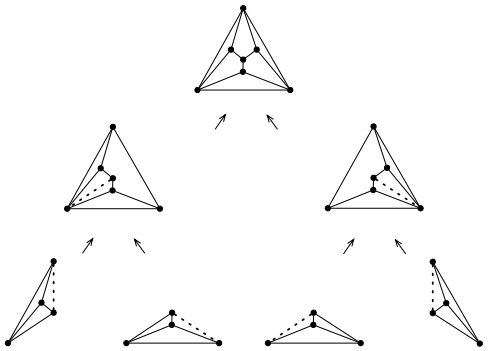
3.1 Tree-Like Hajós Constructions

A Hajós construction can be represented as a directed acyclic graph (DAG) where each non-isomorphic graph in the construction corresponds to a vertex in the DAG. The root vertex of the DAG is a clique in the construction, and directed edges between DAG nodes are the construction operations that generate new graphs. We define a *DAG map* as a diagram of the DAG of a Hajós construction.

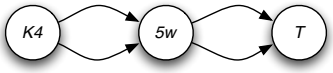
Iwama and Pitassi introduce the *tree-like Hajós calculus* [10] where constructions are structured as a tree. Each graph in a tree-like construction is used only once and must be built independently from cliques. Isomorphic graphs in the construction are not collapsed to a single node. A *tree map* is a DAG map, where the represented DAG is a tree. In an abstract tree representation, each clique graph represents a leaf node in the tree, and the root of the tree is the constructed graph. A regular construction, a tree-like construction, and the DAG and tree maps for the construction are illustrated in Figure 3.1.



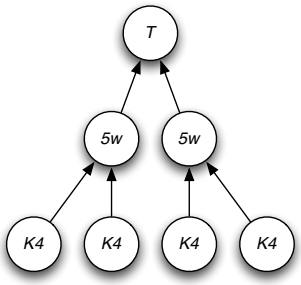
(a) A Hajós construction. Dashed edges represent the edges to be eliminated in the Hajós sum step.



(b) The same Hajós construction operations, expanded to a tree-like construction, with each graph constructed from clique leaf nodes.

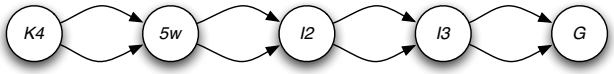


(c) The DAG map of the above construction: K4 is the 4 clique, 5w is the 5-wheel, and T is the final graph.

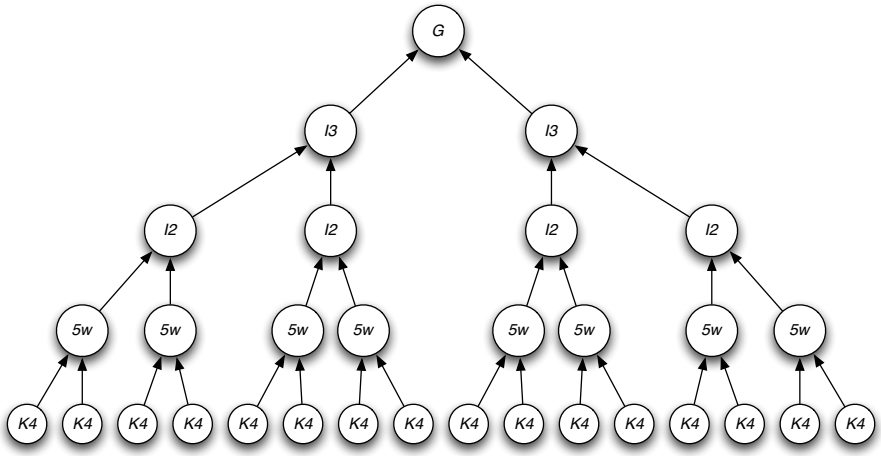


(d) The Tree map of the above construction.

Figure 3.1: An illustration of a Hajós construction, the same construction converted to a tree-like form, a DAG map of the first construction, and a tree map of the second construction.



(a) DAG map



(b) Tree map

Figure 3.2: DAG and tree maps of our construction for the Grötzsch graph of Figure 2.2. I2 and I3 are intermediate graphs.

Two factors contribute to the size difference between regular and tree-like constructions. In tree-like constructions, the same graph can occur many times, and each occurrence contributes to the size of the construction. Additionally, in tree constructions, each occurrence of a graph results in a subtree construction until the leaf node clique graphs are reached. See Figure 3.2, where the DAG map has five vertices, while the tree map has 31.

Iwama and Pitassi show that there are classes of graphs for which tree-like constructions have exponential size, and that the tree-like Hajós calculus cannot polynomially simulate the standard Hajós calculus [10]. This is unexpected, considering that Pitassi and Urquhart's work shows that the Hajós calculus can polynomially simulate Extended Frege proof systems, and tree-like propositional Frege proofs do not lead to significantly larger proofs than non-tree-like proofs [10].

It is straightforward to expand a regular construction into a tree-like construction; however, reducing a tree-like construction to a regular construction is more complicated. This requires identifying duplicate instances of graphs,

so that each unique graph is constructed a single time. Consider a tree-like construction that includes two graphs that are isomorphic. Because the construction is tree-like, the two graphs are generated independently and may have been obtained from non-isomorphic subtree constructions. In reducing such a construction from a tree to a DAG, it is not straightforward to decide which of the two possible subconstructions should be used to obtain the smallest construction. See Figure 3.3.

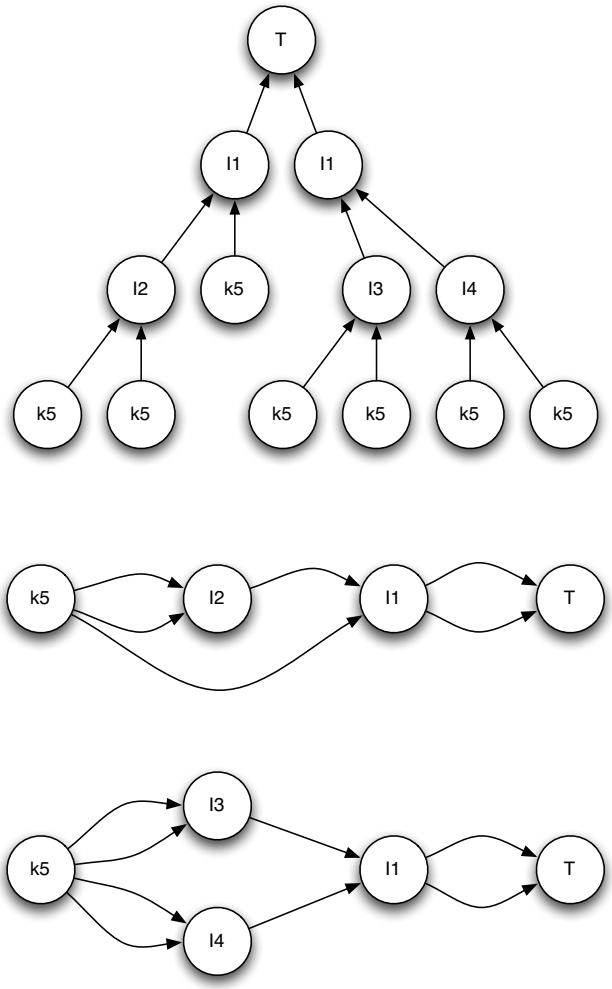


Figure 3.3: Two DAG reductions of the same tree-like construction. The tree has two different subtree constructions of I1. Different reductions yield constructions of different length.

3.2 Hajós' Proof of Completeness

In Section 3.3, we will look at top-down constructions. To understand these constructions, it is helpful to first examine Hajós' proof. The Hajós construction is *complete*: for any k-chromatic graph G, there is a Hajós construction of G from k-cliques. The proof below follows the approaches of Bollobás [2] and Diestel [6].

Theorem 3.2.1. *(Hajós [9]) Any graph of chromatic number greater than or equal to k is Hajós k-constructible.*

Proof. Assume the theorem is false. Let $G = (V, E)$, where $\chi(G) \geq k$, be a counterexample with minimum number of vertices, and among all such graphs, have a maximum number of edges. Because G is a counterexample, G cannot be a k-clique. Also, G cannot be a complete k-partite graph, as that would imply that G has a k-clique as a subgraph, which would imply that G could be Hajós constructed via the supergraph operation. Thus, G has vertices a, b_1, and b_2 where b_1 and b_2 are adjacent, and a is not adjacent to b_1 or b_2.

We show that G can be constructed from two input graphs as follows. Create a graph $G_1 = (V_1, E_1)$ that is disjoint from G and isomorphic to G, with γ_1 being a bijection between V and V_1 as defined by the isomorphism. Add the edge $(\gamma_1(a), \gamma_1(b_1))$ to E_1. Similarly, create a graph $G_2 = (V_2, E_2)$ that is disjoint from G and isomorphic to G, with γ_2 being a bijection between V and V_2. Add the edge $(\gamma_2(a), \gamma_2(b_2))$ to E_2. Note that G_1 and G_2 Hajós construct G: set $G_3 = G_1(\gamma_1(a), \gamma_1(b_1)) +_H G_2(\gamma_2(a), \gamma_2(b_2))$; then, for each $v \in V - \{a\}$, identify the vertices $\gamma_1(v), \gamma_2(v)$; then set the resulting graph as G.

Because G is maximal with regard to edges, neither G_1 nor G_2 are counterexamples, and each has a Hajós construction. Thus, G is Hajós k constructible, and we have a contradiction. □

In addition to being complete, the Hajós construction is also *sound*: any graph G obtained by a Hajós construction from k-cliques has a chromatic number of at least k. If a graph G' is obtained from a graph G via supergraph,

then G' cannot be (k-1)-coloured, as this would define a (k-1)-colouring for G, leading to a contradiction. Similarly, if G' is obtained via a Hajós sum $G' = G_1(x_1y_1) +_H G_2(x_2y_2)$, then in any colouring of G' the vertices y_1 and y_2 would not both have the same colour as the vertex x_1, so this colouring induces a colouring in either G_1 or G_2 and thus uses at least k colours [6].

3.3 Top-Down Constructions

Thus far, we have considered Hajós constructions from the bottom up, building from cliques up to the constructed graph. The proof of completeness for Hajós' theorem includes an alternative top-down view: start with a graph and derive cliques.

We wish to derive Hajós constructions in a top-down manner using the technique of Hajós' proof. We define this as the *add-edge* operation.

1. Given a graph G that is not a complete k-partite graph, select three vertices $\{a, b_1, b_2\}$ from G where b_1 and b_2 are adjacent, and a is not adjacent to b_1 or b_2.

2. Create two new graphs: G_1, a copy of G with the additional edge (a, b_1) and G_2, a copy of G with the additional edge (a, b_2).

See Figure 3.4 for an example of add-edge.

Add-edge allows us to find graphs G_1 and G_2 that can Hajós construct G. We have created two copies of G, each with one additional edge. In reversing this step, we eliminate the introduced edges with the Hajós sum, and we merge the vertices in the two copies of the graph. The edge introduced by the Hajós sum between b_1 of G_1 and b_2 in G_2 merges into the already existing edge (b_1, b_2) in G. Figure 3.5 shows how add-edge can be reversed using Hajós sum and vertex identification.

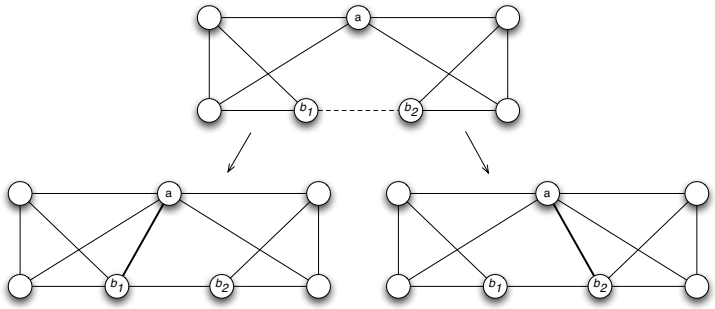


Figure 3.4: The add-edge operation. The top graph has adjacent vertices (b_1, b_2) and a vertex a that is not adjacent to b_1 or b_2. Two copies of the top graph are generated, one with (a, b_1) added and the other with (a, b_2) added.

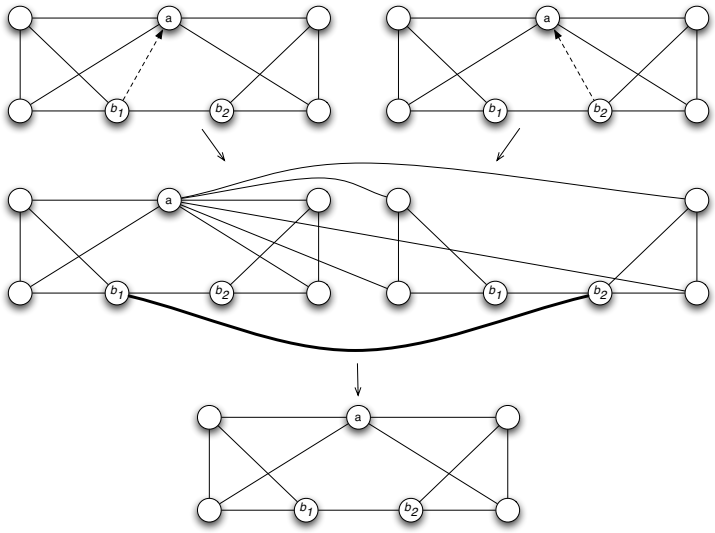


Figure 3.5: Reversing the add-edge operation. A Hajós sum is performed between the two graphs, eliminating (a, b_1) in the left graph and (a, b_2) in the right graph. The a vertices in the input graphs are merged, and an edge between b_1 from the left graph and b_2 from the right graph is added. The bottom graph is then formed by identifying the corresponding vertices from the input graphs.

Add-edge enables a top-down derivation of Hajós constructions: begin with our target graph, and derive a construction in a top-down manner. By recursively adding edges, build a construction tree until each leaf node contains a k-clique subgraph. Then add a final step that constructs each of these leaf nodes by the supergraph rule. See Figure 3.6.

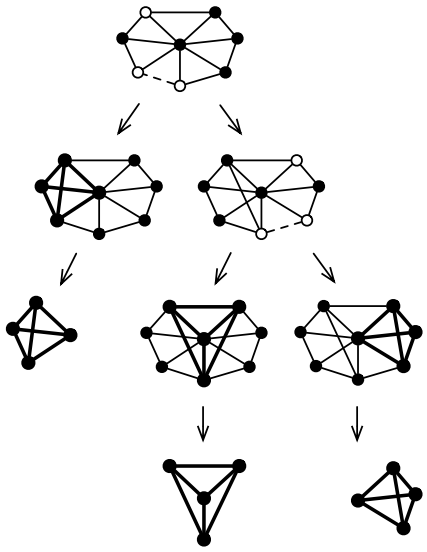
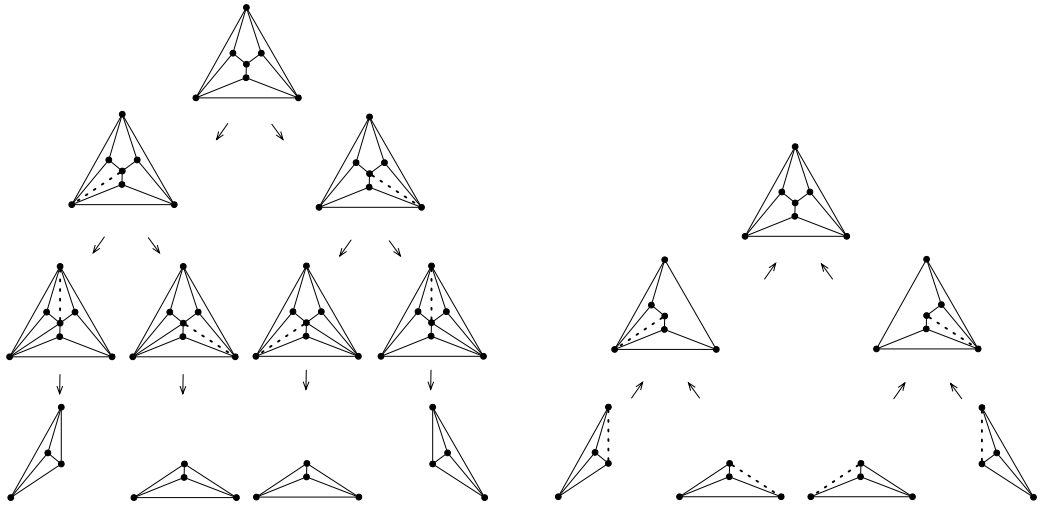


Figure 3.6: Finding a 7-wheel Hajós construction by repeatedly applying add-edge. The construction ends when each leaf graph has a 4-clique subgraph.

These top-down constructions are similar to tree-like constructions with the addition of an initial use of the supergraph rule at the leaf nodes. Additionally, due to the use of add-edge, each intermediate graph in the construction is a strict supergraph of the target.

It is straightforward to convert from a top-down construction to a tree-like construction. As shown in Figure 3.5, each add-edge is a Hajós sum and set of vertex merges. To derive a tree-like construction, we begin with the clique nodes, but we do not apply the initial supergraph operation. At each stage of the construction, we perform the Hajós sum corresponding to reversing the add-edge operation. For vertex merges implied by the add-edge, we merge vertices that exist in both input graphs in the reduced construction, but we ignore merge operations otherwise. See Figure 3.7.

In comparing the sequence of graphs of the reduced construction, notice that each reduced graph is a subgraph of the corresponding graph in the



(a) A top-down add-edge based construction for a seven vertex 4-chromatic graph. Dotted edges represent the additional edge at each step.

(b) A reduced, tree-like construction of the same graph. Dotted edges indicate the edges Hajós sums are performed on.

Figure 3.7: Reducing a construction to required subgraphs.

original construction. The leaf nodes have clique subgraphs. Moving up the construction tree, each edge removed in the construction corresponds to edge removal and vertex merges that define a new corresponding subgraph in the reduced tree. Because the graphs in the reduced construction are Hajós k-constructed subgraphs of the nodes in our original construction, the reduced construction is a valid Hajós construction for G.

When reducing a top-down construction to a tree-like construction, we must consider the possibility that the top-down construction is not minimal. There are non-minimal top-down constructions, where an add-edge operation is applied, but the added edge is not used in any of the leaf cliques of its children. In such cases, this add-edge operation and one of the two branches it creates can be completely removed from the tree, and the resulting tree is still a valid construction.

Ore shows that Hajós' proof, and thus the add-edge operation, can be used to show the completeness of the Ore construction. So any construction generated with our top-down add-edge approach does not require the independent vertex identification operation, and thus is an Ore construction.

Chapter 4

Searching for Hajós Constructions

4.1 Background

As noted by Mansfield and Welsh [14], finding a Hajós construction for a graph can be difficult. In this chapter, we examine some approaches to finding constructions and optimizations to make these approaches practical.

We assume that the graph we wish to construct is k-chromatic and k-critical. To construct a non-critical graph, it suffices to construct a critical subgraph, and then apply the supergraph rule to obtain the non-critical graph.

A natural approach is a bottom-up search: start with cliques and derive graphs via construction operations until the desired graph is discovered. We find that such an approach has difficulties due to the rate that the number of obtainable graphs increases as construction length increases. In Section 2.3, we looked at a bottom-up approach to set a lower limit on the Hajós number of the Grötzsch graph. In Figures 2.3a and 2.3b, we show the two graphs of interest as the output of a single step in a Hajós construction starting from 4-cliques. To explore how the set of obtainable graphs grows as construction length grows, we generate all graphs obtainable from length-two constructions. To do this, we perform all possible Hajós sum operations on the graphs from Figure 2.4, and then we find all graphs derivable via vertex identifications. After removing isomorphic graphs, and eliminating all graphs that have one of the input graphs as a subgraph, we find there are 210 unique graphs obtainable

in two Hajós construction operations.

We encounter difficulty when attempting to enumerate the set of unique graphs obtainable in three operations. Our 210 graphs plus the Hajós graph, the 5-wheel, and the 4-clique form the set of candidates for input to a length-three construction. We find that this set contains 3969 edges. Each pair of edges can be Hajós summed in four ways, and each edge can be summed with itself in three ways. This gives $4 * \binom{3969}{2} + 3 * 3969 = 31509891$ graphs obtainable via Hajós sum. To estimate what would be obtainable via Ore merge for each of the 31 million graphs, we consider all possible vertex merge operations. The number of possible vertex merge operations corresponds to the number of independent sets in each of the graphs. We estimate that the average graph in this set would have 20 vertices and an edge density of 36%. An approximation of the number of independent sets in such a graph is $\sum_{i=2}^{20}(1-0.36)^{\binom{i}{2}} * \binom{20}{i} \approx 987$. This gives us an estimate of $3.1 * 10^{10}$ graphs to consider before isomorphisms are accounted for.

As seen in Chapter 2, the minimum construction for the 11 vertex Grötzsch graph is length four. Based on the size of the set of graphs obtainable from length-three constructions, the bottom-up approach as we have described is not practical for searching for constructions for graphs with minimum construction length greater than three.

4.2 Hajós' Algorithm

In Chapter 3.3, we look at Hajós' proof of completeness and the add-edge operation. Pitassi and Urquhart note that Hajós' proof yields an algorithm for finding constructions [18]. However, this algorithm does not find all constructions, nor does it find minimum constructions.

The add-edge operation allows us to begin with our target graph and derive a construction in a top-down manner. The basic algorithm, shown in Listing 4.1, consists of selecting a triple of vertices that are candidates for the add-edge operation, creating two new graphs using add-edge on the triple, and recursing on those graphs until a clique subgraph is generated.

```
Supergraph_Cons(G):
    if Has_Clique_Subgraph(G):
        return G

    a, b1, b2 = Get_Triple(G)

    G1 = G.copy()
    G1.addEdge((a,b1))

    G2 = G.copy()
    G2.addEdge((a,b2))

    return G, (Supergraph_Cons(G1), Supergraph_Cons(G2))
```

Listing 4.1: Hajós' algorithim.

Hajós' proof shows that this procedure is guaranteed to produce a construction for a graph, but it does not attempt to find small constructions. The top-down add-edge based approach generates tree-like constructions; the number of graphs in the construction doubles at each step, so construction size grows quickly. To get some feeling for the size of constructions produced by an undirected search, we implemented Hajós' algorithm with the vertex triple chosen at random, and then we generated 10,000 constructions of the Grötzsch graph. The shortest construction found had 329 nodes in the tree, and the longest construction had 2145. The mean construction length is 1195.1 graphs with $\sigma = 253.0$. Improvements to this algorithm are needed; the constructions produced are much larger than the minimum length of four that is possible on the Grötzsch graph.

4.3 Searching the Constructions Space

One reason that naively generated constructions can be large is that the algorithm is undirected. We have no criteria for deciding which of the candidate triples of vertices to apply the add-edge operation to. We do not know of any heuristics for making choices amongst the candidate triples that are useful for finding small constructions. So we introduce backtrack search to minimize construction size.

Listing 4.2 shows a brute force algorithm that finds the smallest possible

```
Find_Min_Supergraph_Cons(G):
    if Has_Clique_Subgraph(G):
        return G

    triples = Get_All_Triples(G)

    bestSize = Infinity
    bestCons = None
    for a, b1, b2 in triples:
        G1 = G.copy()
        G1.addEdge((a,b1))

        R1 = Find_Min_Supergraph_Cons(G1)

        G2 = G.copy()
        G2.addEdge((a,b2))

        R2 = Find_Min_Supergraph_Cons(G2)

        size = 1 + Count_Nodes(R1) + Count_Nodes(R2)

        if size < bestSize:
            bestSize = size
            bestCons = R1, R2

    return G, bestCons
```

Listing 4.2: Search for a minimum tree-like supergraph construction.

construction tree that this approach can yield. We have seen that the top-down supergraph constructions directly convert to tree-like constructions, and that the add-edge operation is available under the restricted Ore construction rules, and thus this algorithm produces the minimum tree-like Ore construction for its input graph.

The presented approach repeatedly derives the same construction subtrees unnecessarily. The algorithm considers each candidate triple of vertices independently, but the same edge can occur in multiple triples, and thus the algorithm will recurse on the same edge multiple times. Another consideration is that isomorphic graphs can be generated by the add-edge operation, but the algorithm makes no effort to identify graphs that have been previously constructed.

We will look at more practical approaches in Section 4.5.

4.4 Isomorphisms and Tree-Like Constructions

To find small constructions, we want to convert from top-down tree-like supergraph constructions to DAG constructions as discussed in Section 3.3. Reducing a top-down supergraph construction to a tree-like construction is straightforward. But to further reduce the construction, we must identify isomorphic graphs that occur in our construction multiple times, so that we can collapse them to single nodes in a DAG construction.

The best proven running time for identifying isomorphic graphs is $e^{O(\sqrt{n \log n})}$ [1, 15]. We use the Nauty software package, a high performance graph isomorphism system that gives a canonical labelling for a graph [15]. All isomorphisms of a graph receive the same labelling, allowing us to generate a unique key for any graph. Using this key with a hash table provides a basis for determining if a graph has already been encountered elsewhere. This allows us to further reduce a construction from the tree-like form to a DAG, as discussed in Section 3.1.

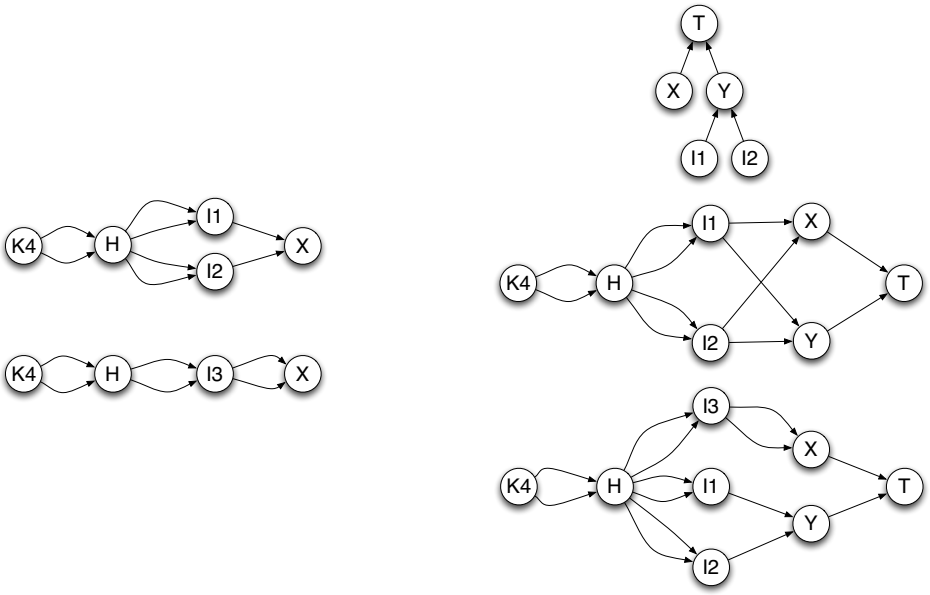
```
# We use a recursive data structure for a construction:
# C.G        - The graph of the head of the construction.
# C.left     - The left branch of the construction.
# C.right    - The right branch of the construction.

seenGraphs = {}

Calc_DAG_Length(C):
    if C == None:
        return 0

    CannonG = Cannonical_Labeling(C.G)
    if CannonG in seenGraphs:
        return 0

    seenGraphs.add(CannonG)
    return 1 + Calc_DAG_Length(C.left) + Calc_DAG_Length(C.right)
```

Listing 4.3: Greedily reduce a tree-like construction to a DAG and return its length.

The algorithm of Listing 4.3 calculates the length of a tree-like construction when it is reduced to a DAG using a greedy approach.

This greedy approach only counts a graph and its subconstruction if an

isomorphism of this graph has not already been encountered. This does not guarantee that the minimum construction size will be returned, as an intermediate graph might appear multiple times with multiple different subconstructions. If our construction contains two different subconstructions for the same graph, our algorithm greedily takes the first subconstruction encountered, even if it is not the smallest. A more complicated approach would search across all possible subconstructions to find the minimum construction.



(a) Two different constructions for the same hypothetical graph X. Considered in isolation the second construction is smaller.

(b) A hypothetical graph T is constructed from graphs X and Y, and Y is constructed from graphs I1 and I2. Considered in the context of a construction containing graph Y, the first construction of X from graphs I1 and I2 results in a smaller DAG, despite the second construction being smaller in isolation.

Figure 4.1: An illustration of the difficulty in optimizing a top-down search.

Notice that it is impossible to calculate the size of a subconstruction in isolation: a graph that appears in a subconstruction may appear elsewhere in the construction tree. Because we do not know the order in which non-isomorphic subconstructions are generated, we do not know how intermediate subconstruction sizes will affect the overall construction size. Thus, we know of no way for a search algorithm to chose an optimal subconstruction without

knowledge of the complete DAG. This is related to the problem of reducing a tree-like construction to a DAG. See Section 3.1. This inability to deal with subconstructions in isolation makes it difficult to apply standard search optimizations such as memoization and branch and bound to our problem. See Figure 4.1 for a hypothetical example.

```
Find_All_Cons(G):
    if Has_Clique_Subgraph(G):
        C = new Construction()
        C.G = G

        return C

    consList = []

    triples = Get_All_Triples(G)

    for a, b1, b2 in triples:
        G1 = G.copy()
        G1.addEdge((a,b1))

        C1 = Find_All_Cons(G1)

        G2 = G.copy()
        G2.addEdge((a,b2))

        C2 = Find_All_Cons(G2)

        for R1 in C1:
            for R2 in C2:
                C = new Construction()
                C.G = G
                C.left = R1
                C.right = R2

                consList.add(C)

    return consList
```

Listing 4.4: Find all possible top-down constructions.

To work around these limitations, we look at searching all possible constructions for a given graph in Listing 4.4.

In Section 3.1, we looked at the relationship between trees and DAGs. It is straightforward to see that a DAG can be expanded to a tree construction, and that this expansion is unique. Consider a DAG that is minimal,

where every graph in the DAG is used in the construction, and where there are no isomorphism between any of the graphs in the DAG. When expanding such a minimal DAG to a tree, every pair of isomorphic graphs in the tree have isomorphic subconstructions—i.e. if the tree contains a graph G with a subconstruction C, then any graph G' that is isomorphic to G has a subconstruction C' where every graph in C' is isomorphic to a graph in C. Note that such a tree construction uniquely reduces back to the same DAG that it was expanded from. Every minimal DAG construction D expands to a unique tree construction T that uniquely reduces to D. The existence of trees with unique DAG reductions implies that we can use our greedy DAG reduction in an exhaustive search. If we consider a tree that uniquely reduces to our desired DAG, our greedy reduction will produce that DAG.

Consider a minimum Ore construction for a graph. Such a construction will not have any pair of isomorphic graphs or subconstructions; otherwise, the construction would not be minimum. Thus, there is a tree construction that uniquely reduces to the DAG for this minimum construction.

The approach taken in Listing 4.5 exhaustively considers all possible minimal constructions under the Ore construction rules. It generates all construction trees and then greedily reduces them to DAGs. Because all trees are considered, a tree that uniquely reduces to a minimum DAG must be considered. Thus, this approach finds the minimum Ore construction for the input

```
Find_Min_Reduced_Cons(G):
    bestSize = Infinity
    bestCons = None

    for C in Find_All_Cons(G):
        reduced = Get_Required_Subgraphs(C)
        size = Calc_DAG_Length(reduced)

        if size < bestSize:
            bestSize = size
            bestCons = reduced

    return bestCons
```

Listing 4.5: Find a minimum length Ore construction.

graph. However, as we will see in Section 6, because there are vertex merge operations that are possible in Hajós constructions that are not allowed in Ore constructions, there are Hajós constructions that are not considered by this algorithm. The algorithm as presented is a purely brute force approach and is impractical even for small graphs. Thus, we need to consider alternatives.

4.5 A Practical Approach

The size of the search space makes the algorithm of Listing 4.5 impractical, even for small graphs. While there is potential for optimizations to be integrated into the approach, our inability to independently calculate the size of a subconstruction makes this difficult. In fact, it may be that the complexity in guaranteeing a minimum construction rivals the complexity of an exhaustive bottom-up search. However, the tools that we have developed in looking at the backwards search have proved useful for an algorithm that explores a limited set of possible constructions. While this is not guaranteed to find minimum constructions, as we will see in Chapter 5, this approach finds constructions for several interesting graphs.

A challenge with the backwards search approach is rapid growth in the size of tree-like constructions. To avoid this, we optimize for constructions that limit this branching. If we can apply an add-edge rule in a manner that does not branch, then we can potentially move towards our goal of generating clique subgraphs without significantly increasing our construction size. Our first strategy is inspired by the construction shown in Figure 2.2. In our Grötzsch graph constructions, due to symmetries in G and the selection of the vertex triple, the generated G_1 and G_2 graphs are isomorphic. For an add-edge step that matches these criteria, our search only needs to find a construction for one new graph. The algorithm in Listing 4.6 can only find constructions that do not branch.

This algorithm does not reduce the construction to its required subgraphs, though that operation could be applied to a completed construction. This limited search is capable of finding the construction of Figure 2.2. We have

```
Find_Nonbranching_Cons(G):
    if Has_Clique_Subgraph(G):
        return G

    triples = Get_All_Triples(G)

    for a, b1, b2 in triples:
        G1 = G.copy()
        G1.addEdge((a,b1))

        G2 = G.copy()
        G2.addEdge((a,b2))

        if Cannonical_Labeling(G1) == Cannonical_Labeling(G2):
            # nodes are isomorphic, search can continue
            # without branching.
            return G, Find_Nonbranching_Cons(G1)

        if Has_Clique_Subgraph(G1):
            # G1 is a leaf node, search can continue
            # on G2 without branching.
            return G, Find_Nonbranching_Cons(G2)

        if Has_Clique_Subgraph(G2):
            # G2 is a leaf node, search can continue
            # on G1 without branching.
            return G, Find_Nonbranching_Cons(G1)

    return None # Failed to find a construction.
```

Listing 4.6: Find a non-branching construction.

found this approach useful in finding constructions for small graphs with high degrees of symmetry, but it fails to find constructions for many graphs. We wish to expand the capability of the search while maintaining bounds on the branching.

In Listing 4.6, we check if a graph represents a leaf node by checking to see if it contains a clique subgraph. To make our search more capable, while keeping our branching to a minimum, we expand our function to check for leaf nodes. As noted in Section 2.3, we know there are two graphs that are directly constructible from 4-cliques. Thus, in addition to our 4-clique, we can search for instances of the 5-wheel or the Hajós graph as subgraphs. However, as we try to increase the number of graphs that we test for, we quickly run into problems. At construction depth of two, we have 210 additional

```
SearchDepth = 3

Leaf_Search(G, depth, seenGraphs):
    if depth == 0:
        return None

    if Has_Clique_Subgraph(G):
        return G

    triples = Get_All_Triples(G)

    for a, b1, b2 in triples:
        G1 = G.copy()
        G1.addEdge((a,b1))

        G2 = G.copy()
        G2.addEdge((a,b2))

        R1 = Leaf_Search(G1, depth-1, seenGraphs)

        newSubgraphs = Used_Subgraphs(R1)

        R2 = Leaf_Search(G2, depth-1, seenGraphs+newSubgraphs)

        # count the size of the reduced construction using
        # isomorphism and our set of previously seen graphs.
        size = Count_Reduced_Size((G, R1, R2), seenGraphs)

        if size < bestSize:
            bestSize = size
            bestCons = R1, R2

    # return the best construction we found,
    # and the set of subgraphs this construction requires
    return bestCons, Used_Subgraphs(bestCons)
```

Listing 4.7: Depth limited leaf search.

graphs to consider for $k = 4$. Similarly, for $k > 4$, the number of candidate graphs that we would wish to consider grows rapidly. In our experiments, this explosion of candidate graphs and the subgraph isomorphism problem being NP-Complete, limits the usefulness of a database for terminating search branches early. Such techniques might be useful for backwards search on very large graphs. However, finding constructions for such graphs seems to be out of reach for the techniques that we know of.

In Listing 4.7, we approach finding leaf nodes in a construction with a depth-limited depth first search. In addition to the depth-limited search, we

use a required subgraph reduction and greedy isomorphism identification. As the main search deepens, we propagate the set of graphs that have already been seen at higher levels of the search. While this doesn't guarantee a minimum construction, it is still useful in reducing construction sizes. In Listing 4.8, we expand on the approach of Listing 4.6, replacing the checks for clique subgraphs with the search function from Listing 4.7.

By limiting the depth that we search to for leaf nodes, we are able to construct some graphs that our non-branching search cannot solve, while being able to control the size of the search space.

Additional optimization opportunities exist—one of which is using graph isomorphism for memoization. This also serves the function of a transposition table and allows for another opportunity to reduce construction sizes based on isomorphism. A second optimization opportunity exists in using a branch and bound approach. For every subgraph that we are constructing, we can look at the smallest construction size encountered so far and determine if the current construction is larger. This approach is limited by our inability to accurately calculate the size of a subconstruction, but it does offer a potential speedup considering that our search is already not guaranteed to find minimum constructions.

```
seenGraphs = {}

Find_Nonbranching_Cons(G):
    if Leaf_Search(G, SearchDepth, seenGraphs):
        return G

    triples = Get_All_Triples(G)

    for a, b1, b2 in triples:
        G1 = G.copy()
        G1.addEdge((a,b1))

        G2 = G.copy()
        G2.addEdge((a,b2))

        if Cannonical_Labeling(G1) == Cannonical_Labeling(G2):
            # nodes are isomorphic, search can continue
            # without branching.
            return G, Find_Nonbranching_Cons(G1)

        C1, newGraphs = Leaf_Search(G1, SearchDepth, seenGraphs)
        if C1:
            # G1 is a leaf node, search can continue
            # on G2 without branching.
            seenGraphs = seenGraphs + newGraphs
            return G, Find_Nonbranching_Cons(G2)

        C2, newGraphs = Leaf_Search(G1, SearchDepth, seenGraphs)
        if C2:
            # G2 is a leaf node, search can continue
            # on G1 without branching.
            seenGraphs = seenGraphs + newGraphs
            return G, Find_Nonbranching_Cons(G1)

    return None # Failed to find a construction.
```

Listing 4.8: Find a construction with branches of specified depth limit using the leaf search of Listing 4.7.

Chapter 5

Example Constructions

In Chapter 2, we show a minimum construction for the Grötzsch graph that we found by hand. We have applied the search ideas described in Chapter 4 to find constructions of additional triangle-free graphs and present those constructions here.

5.1 The Chvátal Graph

The Chvátal graph is a 4-regular, 4-chromatic graph of 12 vertices [4]. See Figure 5.2 for a construction for this graph.

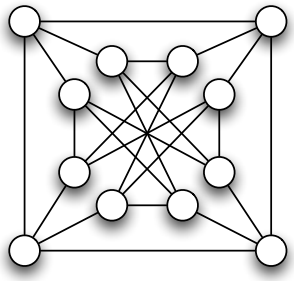


Figure 5.1: The Chvátal graph.

Figure 5.3 shows the DAG map. This DAG is very similar to the DAG of the construction of the Hajós graph, illustrated in Figure 3.2a, though the intermediate and final graphs differ between the two constructions.

By Lemma 2.3.2, there cannot be a length-three construction of the Chvátal graph. Our construction is length four, and thus the Chvátal graph has Hajós number four.

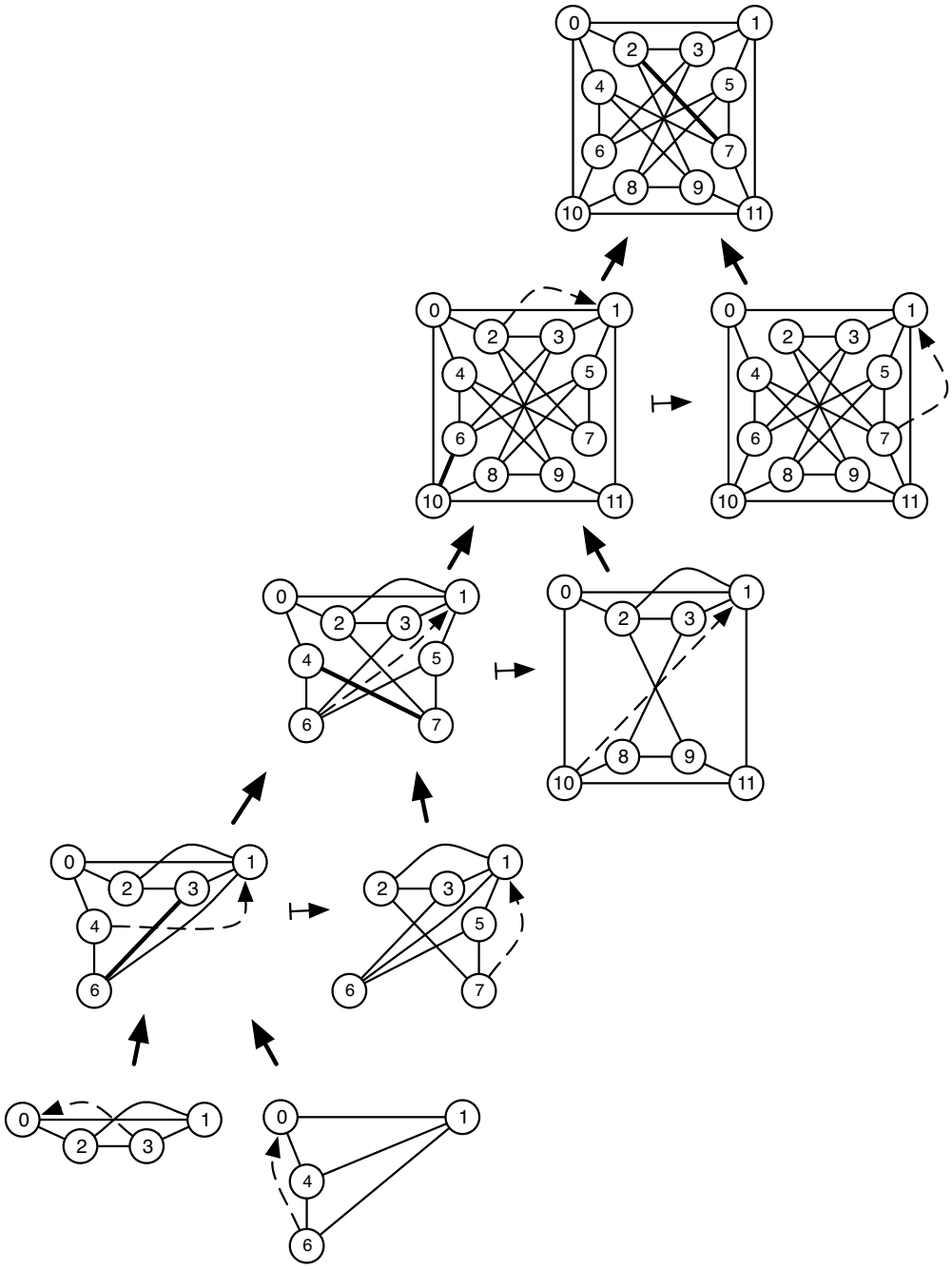


Figure 5.2: A shortest construction of the Chvátal graph, found by our algorithm. This construction has length four. A bold arrow between graphs indicates an Ore merge, where vertices with the same label are identified. The $\mapsto$ arrow indicates an isomorphic copy of a graph with a relabelling of its vertices. For internal graph edges, a bold edge indicates the edge introduced by the Ore merge in constructing the graph, and a dashed arrow indicates the edge that will be deleted in the following Ore merge construction step.

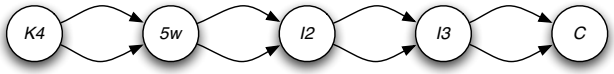


Figure 5.3: The DAG map for Figure 5.2.

5.2 The 13-Cyclotomic Graph

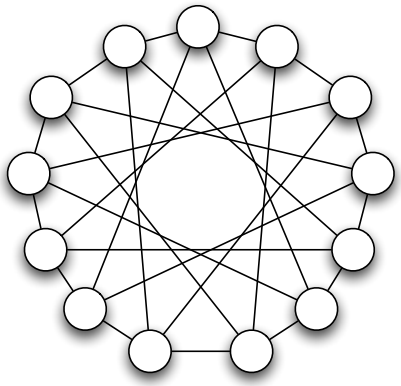


Figure 5.4: The 13-Cyclotomic graph.

The cyclotomic graph of order 13 is another 4-chromatic, triangle-free, and 4-regular graph [21, 20]. We have found a construction of length four. Again by Lemma 2.3.2, we know this is a minimum construction, and that this graph has Hajós number four.

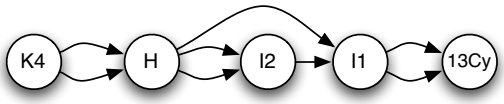


Figure 5.5: A DAG map for the 13-Cyclotomic graph. See Figure 5.6.

The DAG of this construction, see Figure 5.5, has a different form from that of the Grötzsch and Chvátal constructions, though they are the same length.

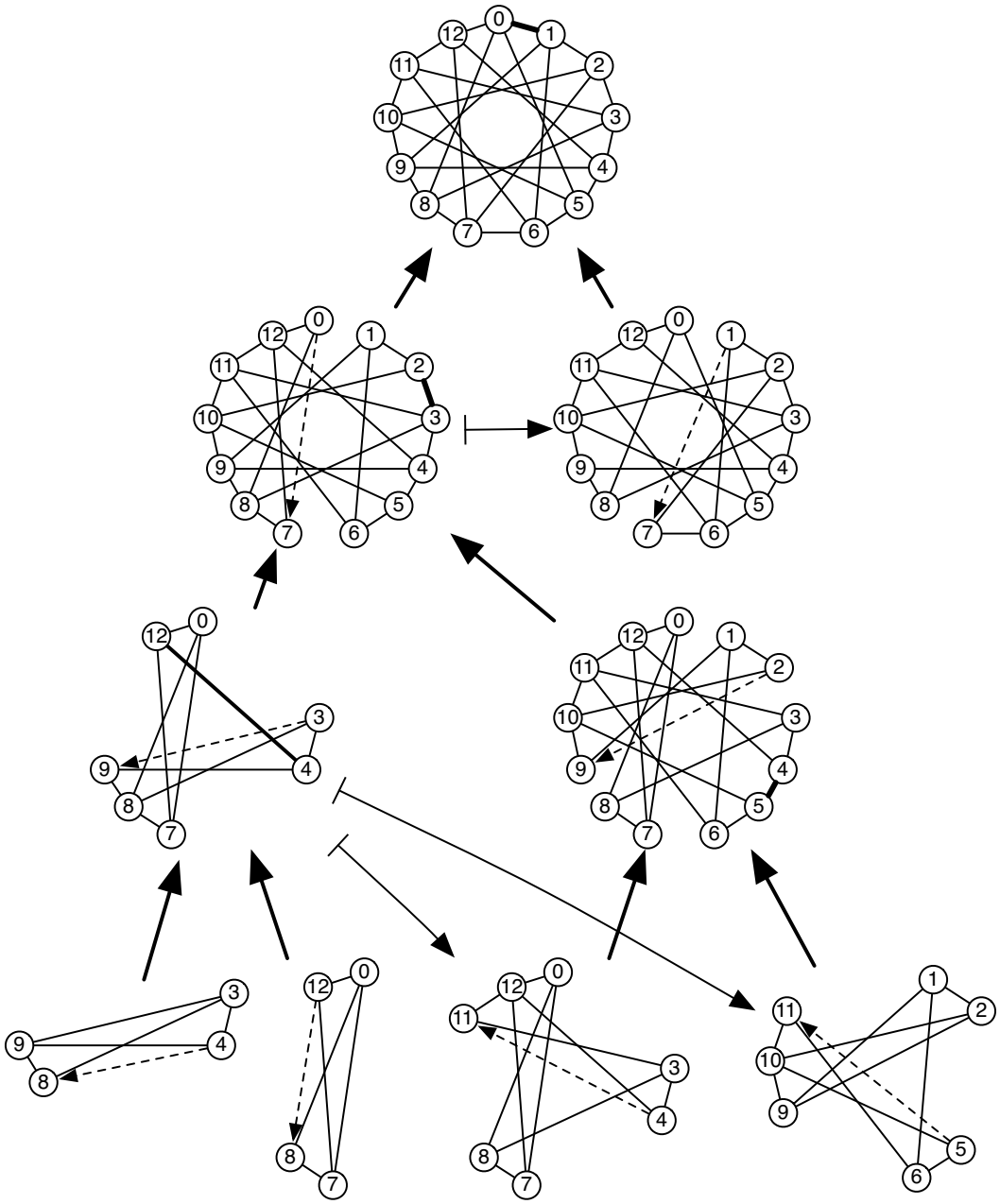


Figure 5.6: A shortest construction of the 13-cyclotomic graph, found by our algorithm. This construction has length four and is illustrated following the same conventions as Figure 5.2.

5.3 The Brinkmann Graph

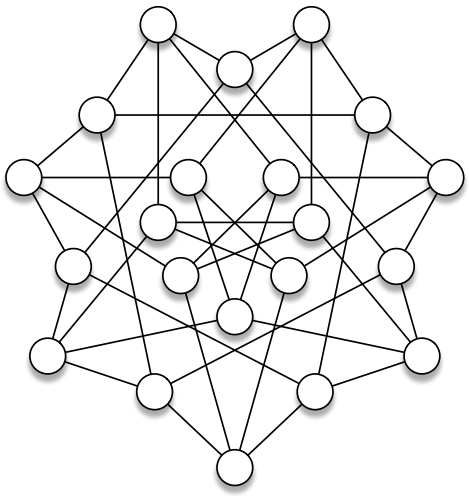


Figure 5.7: The Brinkmann graph.

The Brinkmann graph is a 21 vertex 4-regular, 4-chromatic graph of girth five [3]. So this graph has no triangles or 4-cycles. We show a construction of length 13 for the Brinkmann graph. Due to the length of this construction and the size of this graph, Lemma 2.3.2 does not apply, and we do not know if this is a minimum construction. Thus, we do not know the Hajós number for the Brinkmann graph.

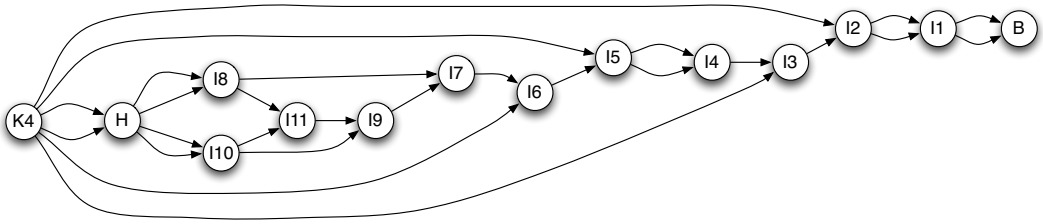
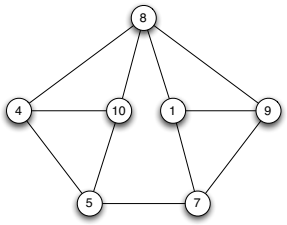


Figure 5.8: A DAG map for the Brinkmann graph, found by our algorithm.

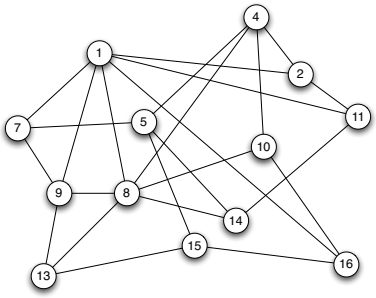
The operations of this construction are described in Table 5.1.

$K4_1$	4-clique labeled: (4, 5, 8, 10)
$K4_2$	4-clique labeled: (1, 7, 8, 9)
H_1	$K4_1(8,5) +_H K4_2(8,7)$
H_2	Relabel H_1 (1, 4, 5, 7, 8, 9, 10) $\mapsto$ (8, 5, 15, 13, 1, 9, 16)
H_3	Relabel H_1 (1, 4, 5, 7, 8, 9, 10) $\mapsto$ (2, 7, 5, 4, 1, 8, 9)
H_4	Relabel H_1 (1, 4, 5, 7, 8, 9, 10) $\mapsto$ (8, 7, 5, 14, 1, 11, 9)
$I10$	$H_1(5,10) +_H H_2(5,16)$
$I10_2$	Relabel $I10$ (1, 2, 8, 9, 11, 13, 14, 17, 19, 20) $\mapsto$ (1, 4, 5, 7, 8, 9, 10, 13, 15, 16)
$I8$	$H_3(8,2) +_H H_3(8,11)$
$I8_2$	Relabel $I8$ (1, 2, 4, 5, 7, 8, 9, 11, 14) $\mapsto$ (8, 4, 2, 20, 9, 1, 13, 10, 16)
$I11$	$I10(5,1) +_H I8(5,9)$
$I9$	$I10_2(1,13) +_H I11(1,9)$
$I7$	$I9(20,8) +_H I8_2(20,13)$
$K4_3$	4-clique labeled: (0, 1, 3, 8)
$I6$	$K4_3(8,0) +_H I7(8,9)$
$I5$	$K4_3(8,0) +_H I6(8,4)$
$I5_2$	Relabel $I5$ (0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 19, 20) $\mapsto$ (2, 1, 0, 11, 4, 10, 16, 14, 20, 5, 3, 17, 8, 12, 7, 13, 19, 9)
$I4$	$I5(1,8) +_H I5_2(1,14)$
$I3$	$I4(1,16) +_H K4_3(1,18)$
$I2$	$I3(1,11) +_H K4_3(1,18)$
$I2_2$	relabel $I2$ (0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20) $\mapsto$ (1, 0, 9, 3, 7, 5, 4, 6, 2, 12, 13, 10, 11, 15, 14, 17, 16, 19, 18, 20)
$I1$	$I2(3,18) +_H I2_2(3,19)$
$I1_2$	Relabel $I1$ (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20) $\mapsto$ (1, 9, 0, 7, 3, 8, 5, 13, 12, 19, 6, 4, 15, 17, 10, 14, 11, 16, 2, 20, 18)
B	$I1(1,0) +_H I1_2(1,9)$

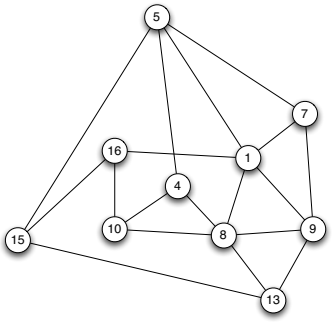
Table 5.1: Operations of a Hajós construction of the Brinkmann graph. Samples of the intermediate graphs are illustrated in figures 5.9 and 5.10. In this construction, each Hajós sum operation is followed by identifying all vertices that share a label.



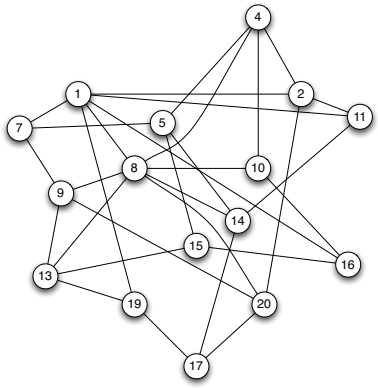
(a) The Hajós graph.



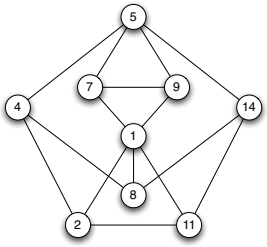
(b) Intermediate graph 1.



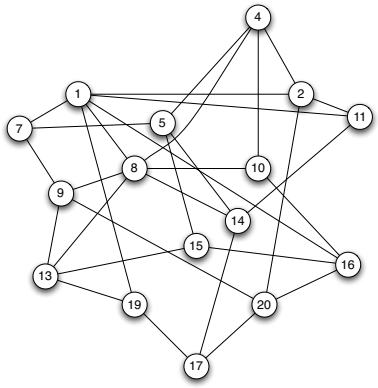
(c) Intermediate graph 2.



(d) Intermediate graph 3.

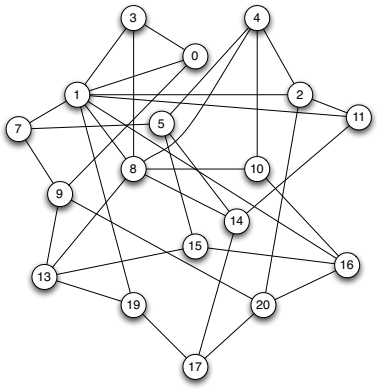


(e) Intermediate graph 4.

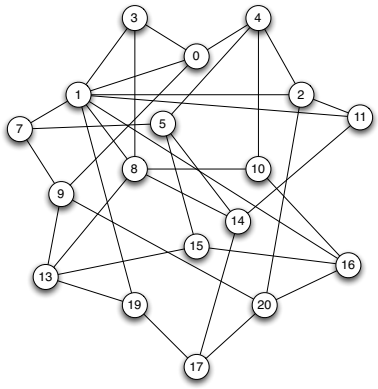


(f) Intermediate graph 5.

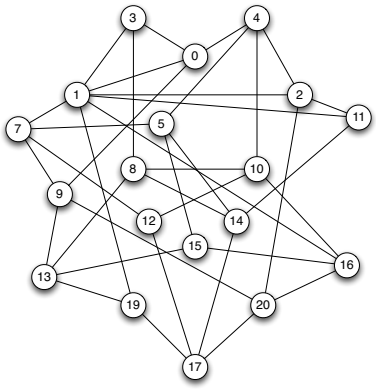
Figure 5.9: Intermediate graphs of our construction of the Brinkmann graph.



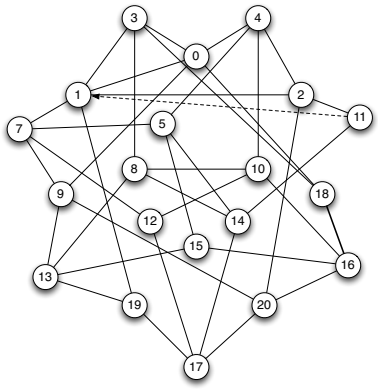
(a) Intermediate graph 6.



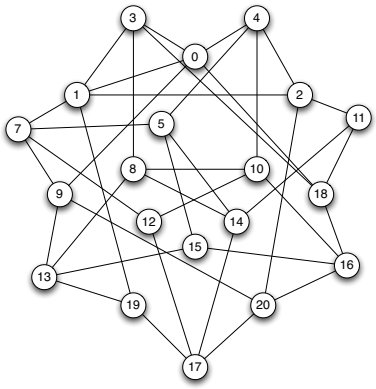
(b) Intermediate graph 7.



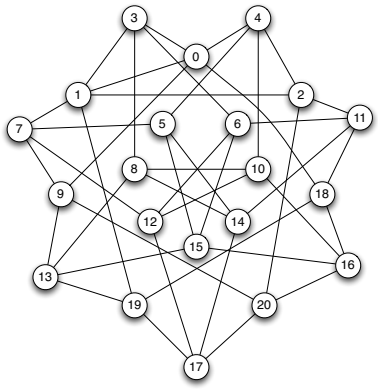
(c) Intermediate graph 8.



(d) Intermediate graph 9.



(e) Intermediate graph 10.



(f) Intermediate graph 11.

Figure 5.10: Intermediate graphs of our construction of the Brinkmann graph, continued.

5.4 The 5-Chromatic Mycielski Graph

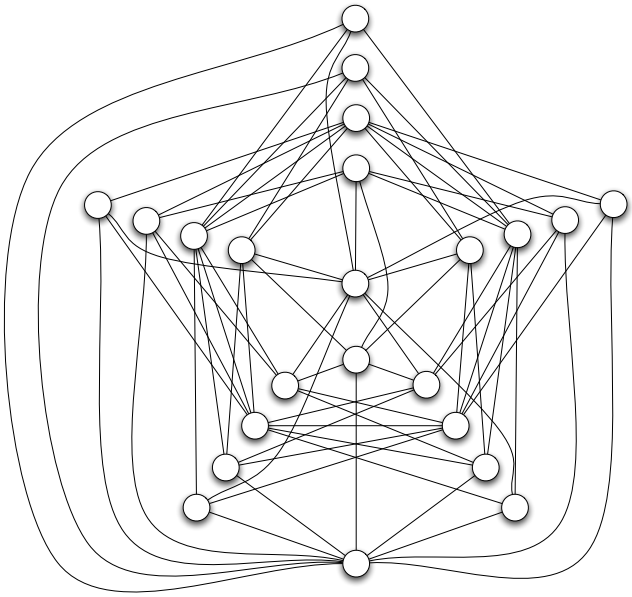


Figure 5.11: M_5: the 5-chromatic Mycielski graph.

Mycielski gives a graph construction showing that a triangle-free graph exists for any chromatic number k [16, 5]. Define M_k as the k-chromatic Mycielski graph, where M_3 is the 5-cycle and M_4 is the Grötzsch graph. Figure 5.11 illustrates M_5. This graph has 23 vertices, is triangle-free, and is 5-chromatic.

Due to the size of the search space, we were unable to find a construction for this graph using the approach described in Section 4.5. Instead, we hand-guided the search based on selecting triples, finding resulting critical subgraphs, and then applying the search algorithm to the resulting graph. We were able to find a length-24 construction. See Figure 5.12 for the DAG map. We do not know the minimum length of a construction for M_5. The operations of this construction are described in Table 5.2 and Table 5.3.

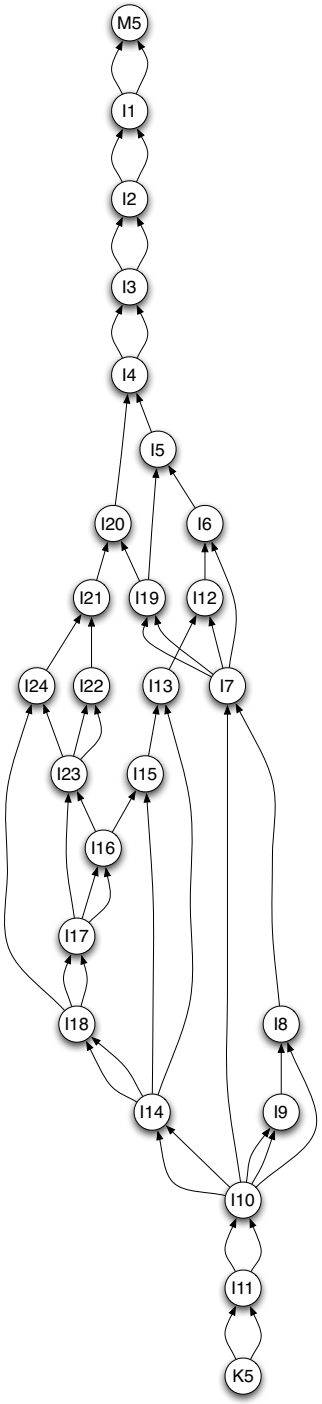


Figure 5.12: DAG map of a length-24 construction of M_5. The top four steps, I4-I3-I2-I1-M5, were found by hand; our algorithm then found the rest of the construction.

$K5_1$	5-clique labeled: (2, 3, 8, 15, 22)
$K5_2$	5-clique labeled: (1, 3, 8, 13, 22)
$I11$	$K5_2(22,1) +_H K5_1(22,2)$
$I11_2$	Relabel $I11$ (1, 2, 3, 8, 13, 15, 22) $\mapsto$ (0, 1, 8, 15, 22, 2, 3)
$I10$	$I11(1,3) +_H I11_2(1,15)$
$I10_2$	Relabel $I10$ (0, 1, 2, 3, 8, 13, 15, 22) $\mapsto$ (12, 5, 15, 8, 22, 21, 0, 7)
$I9$	$I10(8,3) +_H I10_2(8,7)$
$I10_3$	Relabel $I10$ (0, 1, 2, 3, 8, 13, 15, 22) $\mapsto$ (5, 21, 7, 0, 22, 8, 12, 15)
$I8$	$I9(15,2) +_H I10_3(15,12)$
$I10_4$	Relabel $I10$ (0, 1, 2, 3, 8, 13, 15, 22) $\mapsto$ (12, 5, 15, 8, 22, 21, 0, 7)
$I7$	$I8(8,1) +_H I10_4(8,7)$
$I10_5$	Relabel $I10$ (0, 1, 2, 3, 8, 13, 15, 22) $\mapsto$ (7, 3, 13, 8, 22, 15, 1, 0)
$I10_6$	Relabel $I10$ (0, 1, 2, 3, 8, 13, 15, 22) $\mapsto$ (1, 5, 15, 8, 22, 21, 0, 7)
$I14$	$I10_5(8,1) +_H I10_6(8,7)$
$I14_2$	Relabel $I14$ (0, 1, 5, 7, 13, 15) $\mapsto$ (1, 11, 6, 9, 4, 13)
$I14_3$	Relabel $I14$ (0, 1, 5, 7, 8, 13, 15, 21) $\mapsto$ (8, 21, 6, 9, 1, 7, 13, 11)
$I18$	$I14_2(9,1) +_H I14_3(9,8)$
$I18_2$	Relabel $I18$ (1, 3, 4, 6, 7, 8, 9, 11, 13, 21) $\mapsto$ (8, 1, 5, 9, 13, 4, 7, 21, 11, 3)
$I17$	$I18(8,1) +_H I18_2(8,11)$
$I17_2$	Relabel $I17$ (1, 3, 4, 5, 6, 7, 8, 9, 11, 13, 21) $\mapsto$ (5, 8, 11, 4, 7, 15, 9, 13, 1, 21, 3)
$I16$	$I17(8,4) +_H I17_2(8,11)$
$I15$	$I14(22,0) +_H I16(22,9)$
$I13$	$I14(22,0) +_H I15(22,6)$
$I12$	$I7(22,0) +_H I13(22,1)$
$I6$	$I7(22,0) +_H I12(22,4)$
$I7_2$	Relabel $I7$ (0, 1, 2, 8, 12, 13, 21) $\mapsto$ (8, 10, 9, 0, 21, 20, 12)
$I19$	$I7(7,0) +_H I7_2(7,8)$
$I5$	$I6(0,7) +_H I19(0,22)$
$I17_3$	Relabel $I17$ (1, 3, 4, 5, 6, 7, 8, 9, 11, 13, 21) $\mapsto$ (7, 8, 13, 1, 9, 12, 6, 15, 3, 21, 0)
$I16_2$	Relabel $I16$ (1, 3, 4, 5, 6, 7, 8, 9, 11, 13, 15, 21) $\mapsto$ (13, 0, 15, 3, 21, 1, 10, 12, 8, 6, 20, 7)

Table 5.2: Operations of a Hajós construction of M_5. Additional operations are shown in Table 5.3. Each Hajós sum operation is followed by identifying all vertices that share a label.

$I23$	$I17_3(22,9) +_H I16_2(22,10)$
$I23_2$	Relabel $I23$ (0, 1, 3, 6, 7, 12, 13) $\mapsto$ (3, 2, 0, 7, 6, 13, 12)
$I22$	$I23(22,1) +_H I23_2(22,2)$
$I23_3$	Relabel $I23$ (1, 3, 6, 7, 8, 9, 10, 12, 13, 15, 20, 21) $\mapsto$ (12, 10, 1, 8, 3, 6, 2, 15, 7, 20, 17, 13)
$I18_3$	Relabel $I18$ (1, 3, 4, 6, 7, 8, 9, 11, 13, 21) $\mapsto$ (2, 10, 7, 15, 17, 12, 20, 3, 8, 0)
$I24$	$I23_3(22,1) +_H I18_3(22,2)$
$I21$	$I22(22,6) +_H I24(22,10)$
$I20$	$I19(8,0) +_H I21(8,12)$
$I4$	$I5(0,8) +_H I20(0,22)$
$I4_2$	Relabel $I4$ (1, 2, 3, 4, 6, 7, 8, 9, 12, 13, 15, 17, 20) $\mapsto$ (4, 3, 2, 1, 9, 8, 7, 6, 15, 14, 12, 20, 17)
$I3$	$I4_2(22,2) +_H I4(22,3)$
$I3_2$	Relabel $I3$ (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20) $\mapsto$ (3, 4, 0, 1, 2, 8, 9, 5, 6, 7, 14, 15, 11, 12, 13, 20, 18)
$I2$	$I3_2(22,6) +_H I3(22,7)$
$I2_2$	Relabel $I2$ (0, 1, 2, 4, 5, 6, 7, 9, 11, 12, 13, 15, 17, 18, 20) $\mapsto$ (1, 0, 4, 2, 6, 5, 9, 7, 12, 11, 15, 13, 16, 20, 18)
$I1$	$I2_2(22,6) +_H I2(22,5)$
$I1_2$	Relabel $I1$ (1, 2, 3, 4, 6, 7, 8, 9, 12, 13, 14, 15, 17, 18, 20) $\mapsto$ (4, 3, 2, 1, 9, 8, 7, 6, 15, 14, 13, 12, 20, 19, 17)
M_5	$I1_2(22,7) +_H I1(22,8)$

Table 5.3: Operations to Hajós construct M_5, continued. Each Hajós sum operation is followed by identifying all vertices that share a label.

Chapter 6

Hajós and Ore

Hajós' proof that all graphs are Hajós constructible also proves that all graphs are Ore constructible. Urquhart shows a further equivalence between the Hajós operations and the Ore operation: even in the absence of the supergraph rule, any graph that can be constructed via a sequence of Ore merges and vertex identifications can also be constructed via a sequence of only Ore merges [19].

However, there is an important difference between the systems. In a Hajós construction, any two non-adjacent vertices in a graph can be merged at any time, but in an Ore construction a vertex cannot be merged with another vertex in the same graph. Instead, vertices are merged only when two graphs are combined to construct a new graph, and vertices may merge only across the two input graphs. We find that the ability to perform these interior vertex merges gives the Hajós construction additional computational power that distinguishes it from the Ore construction.

To examine the difference between the constructions, we build on the work of Pitassi and Urquhart, who use somewhat different terminology than we have been using so far: a *graph calculus* is a collection of initial graphs and a finite collection of rules that allow the derivation of new graphs [18]. As we note in earlier chapters, both the Hajós and Ore construction systems define a calculus for proving chromatic numbers of graphs. We have thus far been concerned with construction length, but here we look at size. The *size of a graph* is the number of edges in the graph, and the *size of a construction* is the total size of all of the non-isomorphic graphs in the construction. Note

that the size of a construction is polynomially related to its length [18].

Given two graph calculus systems, C_1 and C_2, then C_1 *polynomially simulates* C_2 if there is a polynomial time computable function f, such that for all graphs G, if c_1 is a construction of G in C_2, then $f(s)$ is a construction of G in C_1. C_1 and C_2 are *polynomially equivalent* if C_1 polynomially simulates C_2, and C_2 polynomially simulates C_1.

Pitassi and Urquhart prove that the Hajós calculus is polynomially equivalent in computational complexity to extended Frege proof systems [18]. As part of this work, they define a graph calculus, *HC*, that is similar to the Hajós calculus. HC, like the Hajós calculus, begins with the clique graph and also allows non-adjacent vertices to be merged. But HC replaces Ore merge with *Edge Elimination*.

Edge Elimination: Let G_1 and G_2 be graphs that include the edge (v_1, v_2), where G_1 and G_2 are isomorphic except that G_1 has the additional edge (v_1, v_3) and G_2 has the additional edge (v_2, v_3). Edge Elimination constructs the graph G_3 that is obtained by removing the edge (v_1, v_3) from G_1.

Notice that Edge Elimination can be simulated by Hajós sum and vertex merge operations the same way our add-edge operation, defined in Section 3.3, can be simulated by Hajós construction operations. Our top-down search approach using add-edge produces constructions that map directly to Edge Elimination based HC constructions.

Pitassi and Urquhart also define a proof system they call *HC-*. This system is the same is as HC, except that it does not allow vertex merges.

The full rules of HC- are as follows:

1. Clique base: Start with a k-clique.

2. Edge Elimination: Let G_1 and G_2 be graphs that include the edge (v_1, v_2), where G_1 and G_2 are isomorphic except that G_1 has the additional edge (v_1, v_3) and G_2 has the additional edge (v_2, v_3). Edge Elim-

ination constructs the graph G_3 that is obtained by removing the edge (v_1, v_3) from G_1.

3. Supergraph: Take any existing graph and add any set of vertices and edges.

Pitassi and Urquhart show that graphs with non-polynomial minimum HC- constructions exist. Precisely, they show the following:

Theorem 6.0.1. *(Pitassi and Urquhart [18]) There exists a family of non-3-colorable graphs $\{G_n \mid n \in N\}$ such that for n sufficiently large, any HC- construction of G_n has size at least 2^{n^ϵ}, for some $\epsilon, 0 < \epsilon < 1$.*

We will use this theorem to show that the Ore calculus has non-polynomial constructions by showing that HC- polynomially simulates the Ore calculus. The full rules of the Ore calculus are as follows:

1. Clique base: Start with a k-clique.

2. Ore merge: Given graphs G_1 and G_2 with disjoint vertex sets, an edge (x_1, y_1) of G_1, an edge (x_2, y_2) of G_2, a vertex subset S_1 of $G_1 - x_1$, and a bijection $\gamma : S_2 \leftrightarrow S_1$ where S_2 is a subset of $G_2 - x_2$, the Ore merge $G_3 = O_+(G_1, G_2, (x_1, y_1), (x_2, y_2), S_1, S_2, \gamma)$ is the graph obtained by the Hajós sum $G_3 = G_1(x_1, y_1) +_H G_2(x_2, y_2)$ followed by identifying vertices $v, \gamma(v)$ for all v in S_2.

3. Supergraph: Take any existing graph and add any set of vertices and edges.

We demonstrate the equivalence of HC- and the Ore calculus following the form used by Pitassi and Urquhart to demonstrate the equivalence of HC and the Hajós calculus.

Theorem 6.0.2. *HC- is polynomially equivalent to the Ore Calculus.*

Proof. As both systems include the Clique base and Supergraph rules, it remains to show that Edge Elimination can polynomially simulate Ore merge

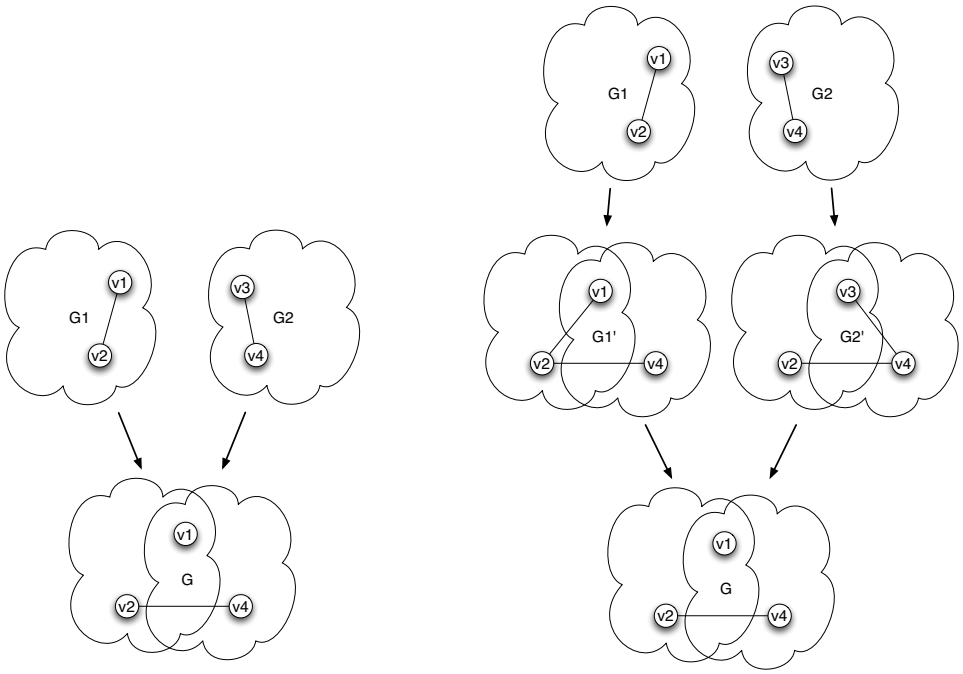
and vice versa. Consider a graph G_1 with the edge (v_1, v_2) and a graph G_2 with the edge (v_3, v_4). Generate a graph G by applying Ore merge on the noted edges of G_1 (v_1, v_2) and G_2 (v_3, v_4). Without loss of generality, we allow the generation of G to include any number of vertex merges, as long as each of the vertices being merged originated in one of the two independent graphs, as the Ore merge step permits. We wish to construct G using the HC- rules. By the constraints of the Ore merge, G must be a supergraph of G_1, with the exception that G does not contain the edge (v_1, v_2). We create a supergraph of G_1, G_1', which is equivalent to G with the additional edge (v_1, v_2). Similarly, we create G_2' as the supergraph of G_2 that is equivalent to G with the addition of the edge (v_3, v_4). Using the Edge Elimination rule of HC-, we can now create G from G_1' and G_2' by eliminating the edge (v_1, v_2) or (v_3, v_4).

Conversely, let graphs G_1 and G_2 be isomorphic, except that G_1 has an additional edge (v_1, v_2), and G_2 has an additional edge (v_1, v_3), where both graphs have the edge (v_2, v_3). Using HC-, let G be a graph generated by Edge Elimination on G_1 and G_2, and is thus identical to G_1 with the edge (v_1, v_2) removed. G is generated from G_1 and G_2 under the Ore construction by eliminating (v_1, v_2) and (v_1, v_3), merging the vertices v_1 across the two graphs and creating an edge between vertex v_2 in G_1 and v_3 in G_2. Then every vertex of G_2 is merged with the corresponding vertex of G_1, as defined by the isomorphism between G_1 and G_2. □

The mapping from Ore merge to Edge Elimination is illustrated in Figure 6.1.

By the equivalence to HC-, there must exist families of graphs which cannot be Ore constructed in size polynomial to the size of the graph. This extension to Pitassi and Urquhat's works implies an important distinction between the Ore construction and the Hajós construction. Graphs exist that have non-polynomial minimum Ore constructions, but it is not known if graphs exist which have non-polynomial minimum Hajós constructions.

This difference between the constructions does not impact our proof that the Hajós number for the Grötzsch, Chvátal, and 13-Cyclotomic graphs is four.



(a) A graph G is obtained via Ore merge on G1 (v1, v2) and G2 (v3, v4) with the identification of some number of vertices.

(b) To construct G under HC-, obtain G1' from G1 and G2' from G2 via Supergraph. G is obtained from G1' and G2' via Edge Elimination.

Figure 6.1: Simulating an Ore merge in HC-.

While each of the constructions we have demonstrated is an Ore construction, our proof that a triangle-free 4-chromatic graph of less that 14 vertices cannot be constructed in length less than four considers the full range of Hajós operations, and holds for both Hajós and Ore constructions.

6.1 Hajós Versus Ore

By the previous theorem, we know there are families of graphs that have minimum Ore constructions of greater than polynomial size, while Hajós constructions may be polynomial. The difference between Pitassi and Urquhart's HC and HC-, and by extension between Hajós and Ore, is the option to merge arbitrary vertices at any point in the construction. That arbitrary merge has two implications for differences in constructions. One is that the depth of an Ore construction is bounded, since Ore constructions are monotonically in-

creasing, whereas a Hajós construction for a graph might include intermediate graphs with more vertices than the target graph. It may be possible that there are graphs with minimum constructions that follow this pattern.

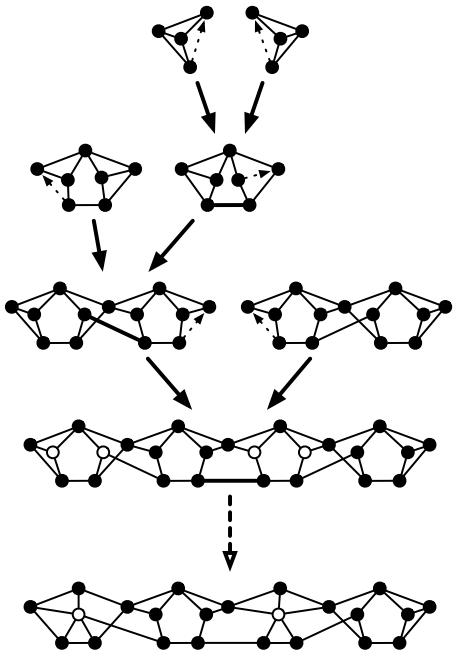
A second implication of the arbitrary merge option is that it can be used to reduce construction sizes by taking advantage of additional isomorphisms between intermediate graphs in a construction. Under the Hajós rules, one can take a copy of an already constructed graph G_1 and identify a vertex to create a new graph, G_2, and use this new graph in the construction without requiring any further construction steps for G_2. Under the Ore rules, this is not allowed. To use G_2 in an Ore construction, it would have to be constructed separately from G_1. See Figure 6.2.

6.2 Top-Down Search with Vertex Merges

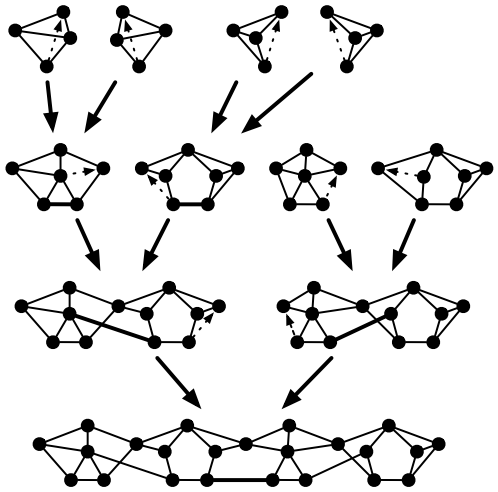
As noted in Chapter 4, the top-down search approach that we have developed derives Ore constructions. Since it is possible that the vertex identification operation may allow for smaller constructions, we consider an extension to the top-down algorithm to account for this difference. To modify our top-down search to cover the full range of Hajós constructions, consider a vertex split operation used in addition to add-edge, as described in Section 3.3.

Vertex Split: Given a k-chromatic graph G, generate G' as follows. Create G' as copy of G and select a vertex v with degree d greater than k. Create two vertices v_1 and v_2. Connect k to d-1 of v's neighbours to v_1, and connect k to d-1 of v's neighbours to v_2, such that all of v's neighbours are connected to one of v_1 or v_2. Remove v from G', and perform a graph-colouring search to attempt to k-1 colour G'. If G' cannot be k-1 coloured, then G' is a valid candidate for a top-down search for a construction of G. Note that G can be Hajós constructed from G' through identifying vertex v_1 with v_2.

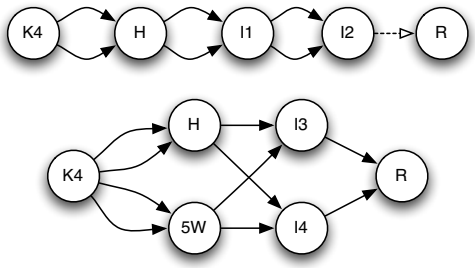
While Vertex Split allows for a top-down search that explores the full possibility space of Hajós constructions, it is not practical. When generating a



(a) A Hajós construction of a graph. The final step involves vertex merges on the vertices marked in white. This results in a final graph with reduced subgraph isomorphism.



(b) An Ore construction of the same graph.



(c) The DAG maps of the two constructions. Note that the Ore construction requires one extra intermediate graph.

Figure 6.2: A demonstration of a smaller Hajós construction and a larger Ore construction for a graph.

new graph via vertex split, one needs to test if the graph is (k-1)-chromatic, which would impose significant overhead on a search algorithm. Additionally, because splitting a vertex will increase the degree of at least some of the neighbouring vertices, the number of graphs that can be generated by a vertex split seems to be unbounded. This indicates that the search space cannot be exhaustively explored without first finding a bound via some other method.

It is unclear if a top-down search including a vertex split would offer any advantages over a bottom-up search. To find minimum Hajós constructions, new techniques and optimizations will be needed.

Chapter 7

A Sequence of (k-1)-Clique-Free k-Chromatic Graphs

Using a brute force search, Jensen and Royle found that there are 56 5-chromatic 4-clique-free graphs with 11 vertices [11]. As an exercise, the reader might like to try to find one of these graphs before reading the rest of this chapter. We have found a sequence that, for any k greater than four, gener-

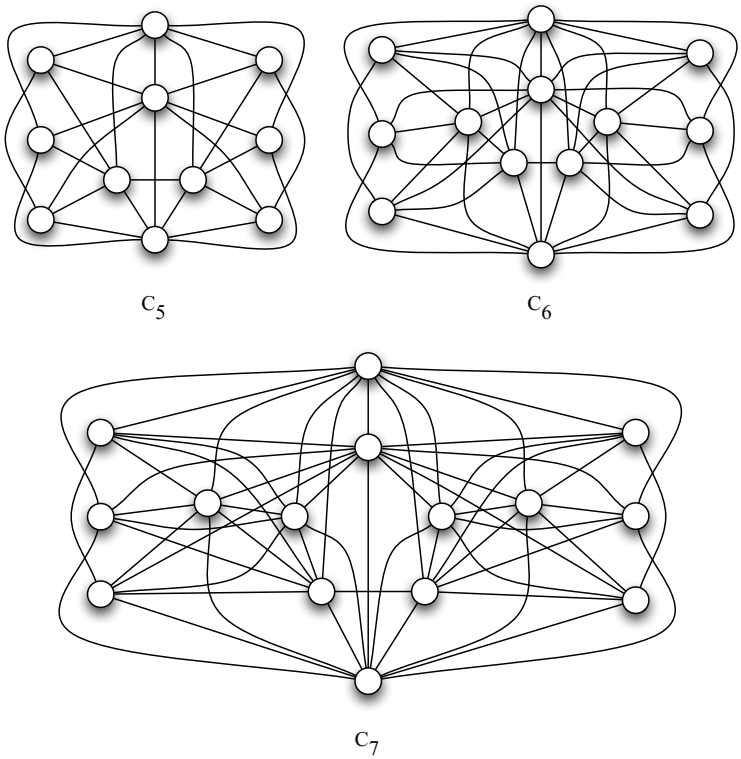


Figure 7.1: The 5-chromatic, 6-chromatic, and 7-chromatic graphs of our sequence.

ates a k-chromatic (k-1)-clique free graph with only $2*k+1$ vertices. C_5 is the 11-vertex graph from our sequence, one of the 56 found by Jensen and Royle. See Figure 7.1. This sequence is defined by a simple set of Hajós construction steps and produces small graphs.

In Section 2.3, we examined a construction that begins with 4-cliques and produces a triangle-free graph in three Hajós sum operations. See Figures 2.4, 2.5, and 2.6. It is straightforward to see how this construction can be extended to construct (k-1)-clique-free graphs by beginning with any k-clique where k is greater than three.

We show a construction that produces (k-1)-clique-free graphs in two Hajós sum operations for k greater than four. In addition to requiring one less sum operation, this construction also produces smaller graphs than the previous approach.

To construct C_k, the k-chromatic graph in our sequence, the following Hajós operations are used:

1. With a k-clique K_1, where $k > 4$, label one vertex v_1 and a second vertex v_2. With a second k-clique, K_2, label one vertex v'_1, and a second vertex v_3.

2. Generate a graph S_1 via Hajós Sum on K_1 (v_1, v_2) $+_H$ K_2 (v'_1, v_3).

3. In S_1, merge two independent vertices not in $\{v_1, v_2, v_3\}$. Perform this step k-3 times. This results in a (k-3)-clique of hub vertices that is fully connected to a set of vertices that form a 5-cycle. Label the two unlabelled vertices of the 5-cycle as v_4 and v_5. Note that this step combined with the previous step is a single Ore merge.

4. Take a copy of S_1, S'_1. Generate S_2 via Hajós sum between a hub edge of S_1 and S'_1. This yields a graph that is (k-1)-clique-free.

5. Create the final graph C_k from S_2 by pairwise merging the two vertices labeled v_5 and the two vertices labeled v_2. This reduces the size of the graph by two vertices, while retaining the (k-1)-clique-free property.

Again note that this step combined with the previous step is a single Ore merge.

These individual steps are illustrated in Figure 7.2 for C_5.

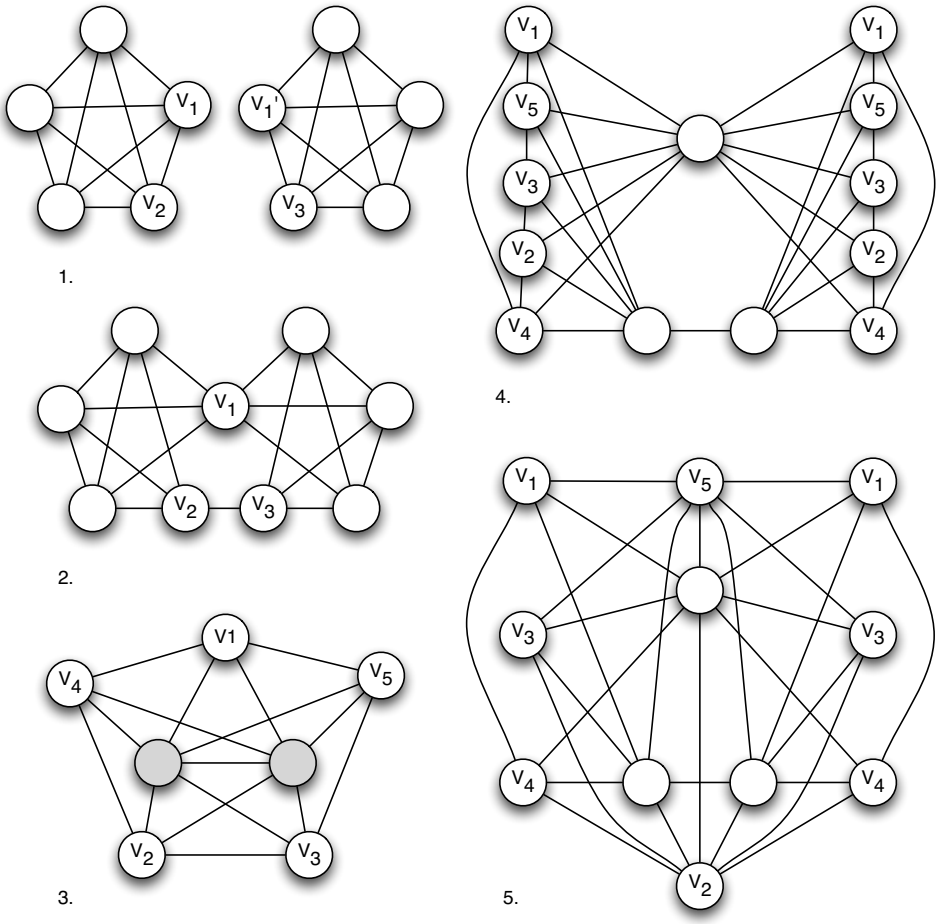


Figure 7.2: Our construction of C_5. In step 3, hub vertices are grey.

We now examine how these construction steps yield a (k-1)-clique-free graph of $2k$+1 vertices. The construction begins with the combination of two k-cliques into a graph with a cycle of five surrounding vertices and k-3 hub vertices. See step three of Figure 7.2. Observe that in these wheel-like graphs, any (k-1)-clique subgraph must include all of the hub vertices. This indicates that a (k-1)-clique-free graph can be generated by eliminating an edge between any pair of hub vertices. We do this by performing a Hajós sum between two copies of the wheel graph. With the initial wheel graphs we have five outer vertices plus k-3 hub vertices, resulting in k+2 vertices. In step

4, Hajós summing the two graphs yields a graph that has $2*(k+2)$-1=$2k+3$ vertices. This Hajós sum eliminates all (k-1)-clique subgraphs.

On the resulting graph, we have a single vertex that was merged as part of the Hajós sum, two sets of vertices that compose the remaining hub vertices from the input graphs, and two sets of vertices that compose the outer five cycles from the input graphs. Select two non-adjacent vertices from one five cycle, and merge them with two non-adjacent vertices from the second five cycle. This yields a k-chromatic, (k-1)-clique-free graph of $2k+1$ vertices.

Our sequence demonstrates that k-chromatic (k-1)-clique-free graphs can be obtained from length-two Hajós constructions for any k greater than four. In contrast, from Section 2.3 we know that a construction of length three is required to obtain a 4-chromatic triangle-free graph.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

Motivated by the aim of finding Hajós constructions for some interesting graphs, we have investigated using computer search to generate Hajós constructions.

Our work uses Hajós' proof to develop a top-down search approach that is effective for finding constructions for small triangle-free graphs. Using this approach, we have found provably minimum constructions for the Grötzsch graph, the Chvátal graph, and the 13-cyclotomic graph. By proving these constructions are minimum, we have demonstrated the Hajós numbers for these graphs to be four. We have also found constructions for the Brinkmann graph and Mycielski's 5-chromatic graph. These are the first published constructions for all of the graphs that we have considered.

Our work also builds on the work of Pitassi and Urquhart, proving that Ore's construction is not polynomially bounded. We examined this difference in power between the Ore construction and the Hajós construction, and we explored the impact this result has on the difficulty in finding minimum Hajós constructions for graphs.

We have also defined a sequence of graphs that are k-chromatic, (k-1)-clique-free of order $2k+1$ that can be constructed in two Ore merge steps.

8.2 Future Work

The barriers that we encountered with optimization in our top-down search approach show some of the difficulty inherent in the problem of finding small Hajós constructions. These challenges present an interesting combinatorial search problem. Despite the difficulty in applying search optimization approaches such as branch and bound, it may be possible to make progress in this area, allowing for a more efficient search, and finding constructions for larger graphs.

We have also found that proving a lower bound on construction lengths is quite difficult. Our work in Section 2.3 provides a lower bound for small triangle-free 4-chromatic graphs. But it is unclear how to approach proving lower bounds, and thus Hajós numbers for larger graphs. We have seen that exhaustively searching for minimum constructions is awkward due to the size of the search spaces. However, further optimization work could allow for exhaustive searches to be performed. This may allow Hajós numbers to be found for more graphs.

Bibliography

[1] L Babai, W M Kantor, and E M Luks. Computational complexity and the classification of finite simple groups. In *24th Annual Symposium on Foundations of Computer Science*, pages 162–171. IEEE, 1983.

[2] B Bollobás. *Modern Graph Theory*. Springer New York, 1998.

[3] G Brinkmann and M Meringer. The smallest 4-regular 4-chromatic graphs with girth 5. *Graph Theory Notes of New York*, 32:40–41, 1997.

[4] V Chvátal. The smallest triangle-free 4-chromatic 4-regular graph. *Journal of Combinatorial Theory*, 9(1):93–94, July 1970.

[5] V Chvátal. The minimality of the Mycielski graph. In *Graphs and combinatorics*, pages 243–246. Springer, 1974.

[6] R Diestel. *Graph Theory*. Springer Verlag, 2006.

[7] M R Garey and D S Johnson. *Computers and Intractability*. A Guide to the Theory of NP-Completeness. W H Freeman & Company, 1979.

[8] H Grötzsch. Ein Dreifarbensatz für dreikreisfreie Netze auf der Kugel. *Wiss. Z. Martin Luther Univ. Halle-Wittenberg, Math. Naturwiss Reihe*, 8(109-120):69, 1959.

[9] G Hajós. Über eine Konstruktion nicht n–färbbarer Graphen, Wiss. *Über eine Konstruktion nicht n–färbbarer Graphen, Wiss*, January 1961.

[10] K Iwama and T Pitassi. Exponential lower bounds for the tree-like Hajós calculus. *Information Processing Letters*, 54(5):289–294, June 1995.

[11] T R Jensen and G F Royle. Small graphs with chromatic number 5: A computer search. *Journal of Graph Theory*, 19(1):107–116, January 1995.

[12] T R Jensen and B Toft. *Graph Coloring Problems*. Wiley, 2011.

[13] S Liu and J Zhang. Using Hajós' Construction to Generate Hard Graph 3-Colorability Instances. In *Artificial Intelligence and Symbolic Computation*, pages 211–225. Springer Berlin, 2006.

[14] A J Mansfield and D J A Welsh. Some colouring problems and their complexity. *North-Holland Mathematics Studies*, 62:159–170, 1982.

[15] B D McKay and A Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, January 2014.

[16] J Mycielski. Sur le coloriage des graphs. *Colloquium Mathematicae*, 2(3):161–162, 1955.

[17] Ø Ore. *The four-color problem*. Academic Press, 1967.

[18] T Pitassi and A Urquhart. The Complexity of the Hajós Calculus. In *Proceedings., 33rd Annual Symposium on Foundations of Computer Science*, pages 187–196. IEEE, 1992.

[19] A Urquhart. The graph constructions of Hajós and Ore. *Journal of Graph Theory*, 26(4):211–215, December 1997.

[20] E R van Dam. *Graphs with few eigenvalues. An interplay between combinatorics and algebra*. PhD thesis, Tilburg University, 1996.

[21] E W Weisstein. Cyclotomic Graph. *Wolfram Mathworld, URL: http://mathworld.wolfram.com/CyclotomicGraph.html*.