Computer search details: Extremal function for $K_9^=$ minors

Martin Rolek*
Department of Mathematics
College of William & Mary
Williamsburg, VA 23185

March 26, 2018

Abstract

In this note, we provide details of the computer searches used to aid the present author in the proof of the extremal function for $K_9^=$ minors [2]. Algorithms used to help prove several lemmas of [2] are given in pseudocode format, and numerical results of the most substantial lemma are also provided (see Table 3.1).

1 Introduction

All graphs considered are simple and finite. We use $V(G)$, $|G|$, $E(G)$, $e(G)$, $\delta(G)$, $\Delta(G)$, and $\overline{G}$ to denote the vertex set, number of vertices, edge set, number of edges, minimum degree, maximum degree, and complement of a graph G, respectively. Given a vertex set $S \subseteq V(G)$, we denote by $G[S]$ the subgraph of G induced by S, and by $G - S$ the subgraph $G[V(G) \setminus S]$ of G. If $S = \{x\}$, we simply write $G - x$ in the latter case. The union (resp. intersection) of two graphs G and H, denoted $G \cup H$ (resp. $G \cap H$), is the graph with vertex set $V(G) \cup V(H)$ (resp. $V(G) \cap V(H)$) and edge set $E(G) \cup E(H)$ (resp. $E(G) \cap E(H)$). The join of two graphs G and H, denoted $G \vee H$, is the graph with vertex set $V(G) \cup V(H)$ and edge set $E(G) \cup E(H) \cup \{uv : u \in V(G), v \in V(H)\}$.

In this note, we will provide details of the computer searches performed for the paper [2]. In [2], the present author proved the extremal function for $K_9^=$ minors, the following.

Theorem 1.1 *If G is a graph with $|G| \geq 8$ and at least $6|G| - 20$ edges, then either $G \geq K_9^=$ or G is a $(K_8, K_{2,2,2,2,2}, 5)$-cockade.*

Proving Theorem 1.1 directly would require an answer to the following question.

Question 1.2 If G is a graph with $8 \leq |G| \leq 11$ and $\delta(G) \geq 5$, does G contain a $K_7^= \cup K_1$ minor?

*E-mail address: msrolek@wm.edu.

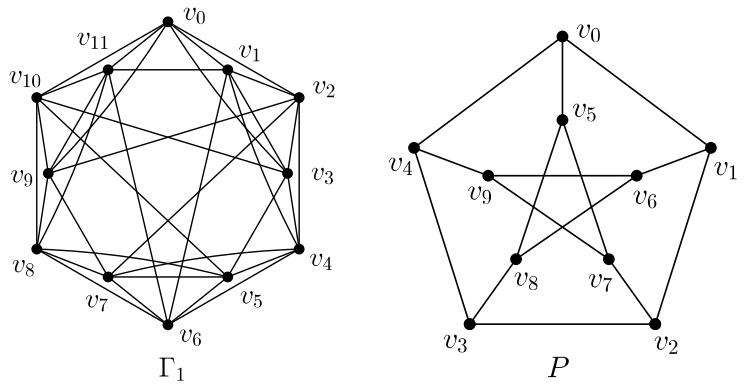

Figure 1.1: The graphs Γ_1 and P of Lemma 1.4.

As we will detail in Section 2, there are at least 3195 graphs G for which the answer to this question is no. In proving Theorem 1.1 directly, these 3195 graphs must be accounted for. Potentially, a set of properties could be found such that any graph satisfying one of these properties can be appropriately handled, and such that every graph satisfies at least one of these properties. This is essentially the method used by Song and Thomas in their proof of the extremal function for K_9 minors [3]. Their corresponding question asked for $K_7 \cup K_1$ minors, but had only 14 graphs for which the answer was no, and they were able to find three such properties (see Lemma 3.7 in [3]). With at least 3195 graphs failing our question, finding such a list of properties seems a daunting task. Instead, the present author proved Theorem 1.1 by first proving the following.

Lemma 1.3 *For any $\epsilon > 0$ and any graph G with $|G| \geq 8$, if $e(G) \geq (6+\epsilon)|G| - 21$, then either G is a $(K_8, K_{2,2,2,2,2}, 5)$-cockade, or $G \geq K_9^=$, or G contains some graph H as a minor such that $e(H) \geq 6|H| - 20$, H is 6-connected, and H has a minimum separating set S such that $H[S]$ is isomorphic to K_6.*

This allows Question 1.2 to be changed so that $9 \leq |G| \leq 12$ and $\delta(G) \geq 6$. While the increase in the maximum number of vertices of our graphs does greatly increase the number of graphs we need to investigate, the increase in the minimum degree more than makes up for this. In fact, with these new conditions, there are only 6 graphs for which the answer is no, summarized as follows.

Lemma 1.4 *If G is a graph with $9 \leq |G| \leq 12$ and $\delta(G) \geq 6$, then either G contains $K_7^= \cup K_1$ as a minor, or G is isomorphic to one of the six graphs $\overline{C_5} \vee \overline{C_4}$, $\overline{C_9}$, $K_{3,3,3}$, $\overline{C_6} \vee \overline{K_3}$, $\overline{P}$, or Γ_1, where $\overline{P}$ is the complement of the Petersen graph P, and Γ_1 and P are as depicted in Figure 1.1. Furthermore, the graphs $\overline{C_5} \vee \overline{C_4}$ and $\overline{C_9}$ contain K_7^- as a minor, and the graphs $K_{3,3,3}$, $\overline{C_6} \vee \overline{K_3}$, $\overline{P}$, and Γ_1 are all edge maximal with respect to not having a $K_7^= \cup K_1$ minor.*

We also use a computer search to help with the final part of Lemma 1.4 by showing that some of the graphs are edge-maximal without a $K_7^= \cup K_1$ minor. We then further use computer searches to aid in the proofs of the following lemmas of [2]. Details of these searches will be provided in Section 4.

Lemma 1.5 *Let e_1, e_2, e_3 be three distinct missing edges of $\overline{P}$ such that no vertex of $\overline{P}$ is incident to every e_i. If either the e_i do not all belong to the same 5 vertex cycle in P or the*

e_i induce a 4 vertex path in P, then $\overline{P} + \{e_1, e_2, e_3\} \geq K_8^=$. Furthermore, the graph obtained from $\overline{P}$ by adding any four missing edges contains $K_8^=$ as a minor.

Lemma 1.6 *Let Γ_1 be as in in Figure 1.1, and let $S = \{v_0, v_4, v_8\}$. Let e_1, e_2 be disjoint missing edges of Γ_1 such that e_1 is incident to exactly one $v \in S$, e_2 is incident to at most one $v \in S$, and no end of e_1 or e_2 is adjacent to both ends of the other edge. Then $\Gamma_1 + e_1 + e_2 \geq K_8^=$.*

2 Finding $K_7^= \cup K_1$ minors - Algorithms

There are two primary algorithms used in our computer search for $K_7^= \cup K_1$ minors. Neither are specific to this particular minor, and so both are presented in generality. Algorithms are presented in pseudocode format. We do not present pseudocode details for any standard graph operations used, such as edge contraction or vertex deletion, as these operations are implemented in a straightforward fashion.

2.1 Edge-minimal

The following Algorithm 1: Edge-minimal determines if a given input graph is edge-minimal with respect to a specified minimum degree. That is, every edge in the graph G must be incident to at least one vertex v such that $d(v) = \delta(G)$.

Algorithm 1: Edge-minimal

Input : G - a graph
δ - minimum degree
Output: True - if G is edge-minimal with respect to minimum degree δ,
False - otherwise

```
for each edge uv ∈ E(G) do
 {
   if d(u) > δ and d(v) > δ then
    {
      return False
    }
 }
return True
```

2.2 $H \cup K_1$ minor

Suppose H is a graph such that if G is a graph with $|G| = |H|$ and $e(G) = e(H)$, then G is isomorphic to H. The following Algorithm 2: $H \cup K_1$ minor allows us to determine if a given graph contains $H \cup K_1$ as a minor. In particular, the following algorithm works for $H \in \{K_t, K_t^-\}$ for a fixed integer t. If we allow $\mathcal{H}$ to be the class consisting of the two nonisomorphic graphs $K_t^=$, then we may also use Algorithm 2 to determine if a graph has an $H \cup K_1$ minor for some $H \in \mathcal{H}$. That is, when searching for a $K_t^=$ subgraph, we do not distinguish between the two nonisomorphic versions of $K_t^=$. We define a function call at the beginning of Algorithm 2, as we intend to call Algorithm 2 recursively.

Algorithm 2: $H \cup K_1$ minor

Input : G - a graph
H - a graph
Output: True - if G contains $H \cup K_1$ as a minor,
False - otherwise

```
H ∪ K_1 Minor(G, H) {
if |G| = |H| + 1 then
 {
   for each vertex v ∈ V(G) do
    {
      if e(G − v) ≥ e(H) then
       {
         return True
       }
    }
   return False
 }

for each edge e ∈ E(G) do
 {
   if H ∪ K_1 Minor(G/e, H) = True then
    {
      return True
    }
 }

return False }
```

The first for loop determines if a minor exists by simply counting edges when the number of vertices of G precludes any further edge contractions. The second for loop iterates through all possible sequences of edge contractions and recursively checks for an $H \cup K_1$ minor. The second loop will exit Algorithm 2 once any one such sequence is found resulting in a $K_7^{=} \cup K_1$ minor. The implementation of Algorithm 2 is a naive, brute-force method with no attempt at any sort of optimization. Some optimization should be possible if the sequences of edge-contractions are chosen in a more intelligent fashion. For example, prioritizing the selection of edges e whose ends have the fewest possible common neighbors would produce graphs G/e with more edges, thus increasing the liklihood of finding an $H \cup K_1$ sooner in the search. We remark that the runtime of Algorithm 2: $H \cup K_1$ Minor on the entire search space for Lemma 1.4 is shorter than the time necessary to implement such an optimization. However, for any future searches on larger graphs, such an optimization would certainly be beneficial.

3 Finding $K_7^{=} \cup K_1$ minors - Search details

All graphs are generated using McKay's Nauty program [1]. When searching for a $K_7^{=} \cup K_1$ minor, we need only check graphs which are edge-minimal with respect to the conditions of Question 1.2 or Lemma 1.4, since every graph meeting the conditions contains an edge-

Question 1.2: $\delta(G) = 5$

$\lvert G\rvert$	$\max(e(G))$	# Generated Graphs	# Edge-Minimal Graphs	# Graphs with no $K_7^= \cup K_1$ minor
8	21	11	7	7
9	24	123	32	32
10	28	16974	483	465
11	32	3592401	10877	2691
	Total	3609509	11399	3195

Lemma 1.4: $\delta(G) = 6$

$\lvert G\rvert$	$\max(e(G))$	# Generated Graphs	# Edge-Minimal Graphs	# Graphs with no $K_7^= \cup K_1$ minor
9	28	15	9	4
10	32	602	71	1
11	36	125296	2516	0
12	40	48948186	176323	1
	Total	49074099	178919	6

Table 3.1: Computer search results for Question 1.2 and Lemma 1.4.

minimal graph meeting the conditions as a minor. Nauty allows a specified minimum degree during graph generation, but does not offer control over edge-minimality with respect to this minimum degree, necessitating the use of Algorithm 1: Edge-Minimal. The graph generation alone can be a time-consuming process, so we generate graphs with the edge-minimal condition in mind. If $\delta(G) = \delta$, then every vertex $v \in V(G)$ with $d(v) > \delta$ can be adjacent only to vertices of degree δ in our edge-minimal graph. Let n_δ denote the number of vertices in $V(G)$ with degree δ. Then $2e(G) \leq \delta n_\delta + n_\delta(|G| - n_\delta)$. After generating the graphs, we use Algorithm 1: Edge-Minimal to pare the list to only those graphs edge-minimal with respect to the minimum degree. Then we use Algorithm 2: $H \cup K_1$ minor to determine the graphs without a $K_7^= \cup K_1$ minor. Table 3.1 summarizes the results of this computer search.

4 Handling counterexamples to Lemma 1.4

In this section, we detail a further three algorithms used to help account for the counterexamples to Lemma 1.4 in the proof of Lemma 1.3.

4.1 Edge-maximal

We say that a graph G is edge-maximal with respect to not having a property $\mathcal{P}$ if G does not have property $\mathcal{P}$, but for any $e \in E(\overline{G})$, $G + e$ has property $\mathcal{P}$. The following Algorithm 3: Edge-maximal is used to verify the final part of the statement of Lemma 1.4. That is, if $G \in \{K_{3,3,3}, \overline{C_6} \vee \overline{K_3}, \overline{P}, \Gamma_1\}$, then $G + e$ has a $K_7^= \cup K_1$ minor for every $e \in E(\overline{G})$. This algorithm utilizes Algorithm 2: $H \cup K_1$ minor.

Algorithm 3: Edge-maximal

Input : G - a graph
Output: For every $e \in E(\overline{G})$, prints whether $G + e$ has a $K_7^= \cup K_1$ minor or not

for *each edge* $uv \in E(\overline{G})$ **do**
{
  if $H \cup K_1$ `Minor(`$G + uv, K_7^=$`)` $=$ *True* **then**
  {
    print "uv: Yes"
  }
  else
  {
    print "uv: No"
  }
}
return

4.2 Lemma 1.5

To prove Lemma 1.5, we wish to find a condition that three edges $e_1, e_2, e_3 \in E(P)$ must satisfy such that $\overline{P} + \{e_1, e_2, e_3\} \geq K_8^=$. By recalling that the Petersen graph is edge-transitive, we may assume that one edge e_1 is fixed. Then we choose 2 of the remaining 14 edges to be e_2, e_3. The following Algorithm 4: Petersen checks each of the $\binom{14}{2} = 91$ pairs. Manual inspection of the 14 pairs which result in a graph without a $K_7^= \cup K_1$ minor reveals the conditions as stated in Lemma 1.5. Again, Algorithm 4 utilizes Algorithm 2: $H \cup K_1$ minor.

Algorithm 4: Petersen

Output: Prints whether $\overline{P} + \{e_1, e_2, e_3\}$ has a $K_7^= \cup K_1$ minor or not

fix $e_1 \in E(P)$
for *each edge* $e_2 \in E(P - e_1)$ **do**
{
  for *each edge* $e_3 \in E(P - \{e_1, e_2\})$ **do**
  {
    if $H \cup K_1$ `Minor(`$\overline{P} + \{e_1, e_2, e_3\}, K_7^=$`)` $=$ *True* **then**
    {
      print "e_2, e_3: Yes"
    }
    else
    {
      print "e_2, e_3: No"
    }
  }
}
return

4.3 Lemma 1.6

The following Algorithm 5: Γ_1 helps us to prove Lemma 1.6. Note that $e(\overline{\Gamma_1}) = 30$, so there are $\binom{30}{2} = 435$ pairs of missing edges, and Algorithm 5 checks all of these pairs using Algorithm 2: $H \cup K_1$ minor. We used non-exhaustive manual inspection to conjecture the conditions of Lemma 1.6. To fully verify, we exploit the symmetry of Γ_1 and fix one end of v_1, leaving only 30 pairs whose results we needed to check.

Algorithm 5: Γ_1

Output: Prints whether $\Gamma_1 + \{e_1, e_2\}$ has a $K_7^= \cup K_1$ minor or not

for *each edge* $e_1 \in E(\overline{\Gamma_1})$ **do**
 {
  for *each edge* $e_2 \in E(\overline{\Gamma_1} - e_1)$ **do**
   {
    if $H \cup K_1$ `Minor(`$\Gamma_1 + \{e_1, e_2\}, K_7^=$`)` $=$ *True* **then { print** "e_1, e_2: Yes"; **}**
    else { print "e_1, e_2: No"; **}**
   }
 }
return

References

[1] B. McKay, http://users.cecs.anu.edu.au/∼bdm/nauty/

[2] M. Rolek, "The extremal function for $K_9^=$ minors," *submitted.*

[3] Z-X. Song and R. Thomas, "The extremal function for K_9 minors," *J. Combin. Theory, Ser. B.* **96** (2006), 240–252.