On Sigbovik Paper Maximization

Josh Abrams

March 2021

1 Introduction

The subject of Sigbovik Paper Minimization (i.e. "What is the shortest possible paper that could be submitted to and accepted by Sigbovik?") has been studied quite thoroughly in recent years, producing many great theoretical and practical successes [5, 2, 3, 4, 7, 6].

However, a natural related problem that has received less attention is that of Sigbovik Paper Maximization.

We believe that the reason for this knowledge gap may be three-fold:

1. Current state-of-the-art, efficient paper construction algorithms are quite biased towards short output sizes.
2. Whereas the problem of paper minimization is rather straightforward, there are many different metrics for paper maximization.
3. Sigbovik is in the pocket of big small [8].

Assuming you are reading this paper in an official copy of the Sigbovik proceedings, we can safely eliminate (3) as a possibility (and if you are reading this paper because millions of copies are being dropped from overhead blimps, then the revolution is already upon us). Thus, our focus here will be on addressing the first two potential issues.

We present a few metrics that have proven useful to the theory of paper maximization, as well as several novel, efficient algorithms for generating maximal Sigbovik papers while using as little work and intelligent thought as humanly (or computationally with the use of state-of-the-art artificial stupidity algorithms) possible.

2 Quick Formalisms

As our reader has probably already figured out, the hardness of this problem does not come from the difficulty of making a paper arbitrarily long. On any

computer with a reasonable amount of RAM, we could easily write a program to do something like this:

Listing 1: Generating a really long paper

```
really_long = "A" * (1 << 30)

with open("paper.txt", "w") as f:
    f.write(really_long)
```

Which gives us 2^{30} characters, which (based on some disreputable sources on the average number of characters per page) gives us a page count of $\frac{2^{30}}{3000} \geq 350,000$ pages. So why not just do this?

The answer, of course, is that for some as yet undiscovered reason, SIGBOVIK reviewers do not like reading incredibly long papers with little to no useful information. Thus, our problem is not just to make the paper as large (by some metric) as possible, it is also to come up with a method for ensuring the paper has some non-negligible chance of being accepted.

To formalize this, we introduce the notion of "Big O...MFG." A paper production algorithm, A, with paper-size metric, μ, is in $OMFG(f, \mu)$ if using A to produce a paper, P such that $\mu(P) \geq x$ has probability at most $c \cdot f(x)$ of being accepted to SIGBOVIK, for some constant c.

For example, if we take our metric, μ to be page count and f to be 1 for $x \in [0, 3]$ and 0 otherwise, we theorize that the algorithm shown above (just repeat a single character) is $OMFG(f, \mu)$. Perhaps with a catchy title, we could get away with simple repetition for a few pages, but we will certainly not be breaking any records.

There have been several deep theoretical results in this realm. For example:

Theorem 1 (The Lorax Impossibility Theorem)**.** *If your metric, μ, is some function f of the page count, with $f \in \omega(1)$, then for any algorithm, A, we have $A \in OMFG(g, \mu) \implies g(x) = 0$ for all x greater than some k^*.*

In other words, if the page count must grow to arbitrary size as μ increases, we will eventually hit a point where acceptance is impossible.

The original proof of this theorem was as follows:

Proof. Every year, the SIGBOVIK proceedings are printed. The number of atoms in the universe is finite. Thus, for a sufficiently long paper, we would use up all the trees in the universe and then be unable to print the proceedings. □

Some of the above assumptions may not be well-founded. However, there are extensions to the proof that are robust to the possibility of a fully-digitized release.

It is worth noting that research in Paper Maximization Complexity Theory is still in its infancy and big OMFG bounds can be rather tricky to prove. Thus, it

is unlikely that we will be able to provide strong upper bounds on the algorithms that appear later in this paper.

3 Metrics for Maximization

As alluded to in the previous sections, there are many different ways to measure largeness of a paper that can change our approach. A few of the common metrics are listed below:

3.1 Page Length

The most traditional measure of a paper's size is the number of pages it occupies. Some advantages of this metric are that it is rather easy to measure and allows for some trivial, yet performant algorithms.

However, it can be very difficult to achieve non-negligible acceptance probabilities for even modest lengths (the most successful algorithm has been the incredibly inelegant: "Actually Put Some Thought and wOrk into REsearching Something" (APSTORES)). Furthermore, we know by the Lorax Impossibility Theorem that there are strict limits on our performance by this metric.

3.2 Character Count and Word Count

These two metrics should also be quite familiar and, in most cases, are intimately tied to page count. They measure the number of symbols from some alphabet Σ that appear in the paper, and the number of strings from Σ^* delimited by spaces, respectively.

Thus, while the algorithm in Listing 1 had a respectable character count of 2^{30}, most text editors would rate its word count as a measly 1.

These measures are useful because they appeal to our natural understandings of size but allow for algorithms that can produce impressive sizes while maintaining a decent chance of acceptance, as we will see.

3.3 Information Content/Density

Finally, some readers might have been especially unimpressed with Listing 1 because the output paper was incredibly compressible. Thus, another approach to measuring paper largeness, is to think about the paper's Kolmogorov complexity or its size in bits when using a Shannon optimal coding scheme.

4 Algorithms for Maximization

At last, the background is done and we can start proposing algorithms and breaking records. What follows are some of our most promising candidates, some of which have been run to make this paper quite large by some metrics.

4.1 Polyplagiarize

Polyplagiarize, or the Page Pirate Algorithm, is a method for easily making incremental improvements on maximal paper size while maintaining reasonable acceptance probability by plagiarizing existing work.

Specifically, we find a long existing work, copy it, and then resubmit it under our name with some small extensions.

Algorithm 1 The Poly-Plagiarize, or Page Pirate Algorithm

1: **procedure** PP
2: Look through the SIGBOVIK archives for the longest paper so far.
3: Add a page of acknowledgements or extra references.
4: Change the author to your name.
5: Potentially make some trivial modifications.
6: Resubmit the paper.
7: **end procedure**

An obvious advantage of this algorithm is the idea that "if they took it once, they'll probably take it again," so acceptance probability is rather high.

However, the method prevents us from growing by much more than a few pages per year. Furthermore, there is the added risk that if you're caught violating academic integrity, the 15-213 course staff will come and confiscate your diploma and your soul.

One way to avoid this fate is to try using the related algorithm of Reference Unpacking:

Algorithm 2 The Reference Unpacking Algorithm

1: **procedure** RefUnpack
2: Write some trivial paper, P, that references a much longer paper, P'.
3: Instead of using standard citation practices to reference P', simply copy the entire text.
4: Add some words to make it sort of make sense.
5: Submit and pray.
6: **end procedure**

For example, suppose we wanted to write a paper titled, "On the Coolness of Tiny SIGBOVIK Papers," whose text was something to the effect of:

> In 2019, Patrick Lin published a paper refuting the tinyness lower bounds proposed by Jones in the same year [4, 3]. This was pretty cool.

Running the Reference Unpacking Algorithm, we would end up with:

> In 2019, Patrick Lin published his paper, "No, this is the tiniest SIGBOVIK paper," which read: "Eat your heart out Mitchell," in response to Mitchell Jones' paper from the same year, "Is this the tiniest SIGBOVIK paper ever?" whose text was just "'I have discovered a truly remarkable proof of this theorem which this margin is too small to contain.' – Some lawyer in the 1600's." This was pretty cool.

Thus, we have made some pretty significant length increases using only references to small papers. And, our souls are safe since everything is properly attributed.

However, we do have much lower probability of acceptance when using this method, which is why the prayer in the final step of the algorithm is particularly necessary. However, the final step does motivate another approach, which we discuss next.

4.2 Maximization by Divine Intervention

With the right choice of text, we believe we can get massive paper sizes and high acceptance probabilities using very little work. Specifically, we do the following:

Algorithm 3 God's Algorithm

1: **procedure** GodAlgorithm(n)
2: Download a full text of the holy Bible
3: Add n copies of the text to a text file.
4: Add some kind of weird title and submit.
5: Make it very clear to all that it would be heretical not to accept the paper, and that a rejection would be punishable by a smiting.
6: **end procedure**

This allows us to produce papers of page count on the order of $1000n$ and word count on the order of $700000n$, which would likely beat out any previous contenders. The only drawback is that heretical ideology is quickly growing in popularity, so effectiveness may decline over time.

4.3 RandoSpam

While the Bible can get us great paper sizes and decent acceptance probabilities under the page/word count metrics, there are some unfortunate issues that might persuade us to use other schemes for the complexity metric. Specifically,

1. The Kolmogorov Complexity of the Bible is very low. It can be proven constructively that it is, in fact, at most 97 bytes since the following script should print it out:

   Listing 2: Generating the Bible

   ```
   wget https://raw.githubusercontent.com/mxw/grmr/
        master/src/finaltests/bible.txt
        && cat bible.txt
   ```

   (NOTE: The line breaks were added just to make this fit nicely in the paper and should not count towards the byte total).

2. Even if it weren't easy to output a Bible, the fact that it's written in English cripples its information density—by some estimates, English text is readily compressible by factor of between 4 and 8 (assuming an ASCII encoding).

If we really want to get the best mileage out of our symbols, the best solution is to just use a string of random characters. This ensures both high Kolmogorov complexity and low compressibility.

But there is a problem. Since SIGBOVIK reviewers are servants of "The Man," they most likely find highly entropic texts to be anarchic and therefore unsettling. Thus, we would be unlikely to get this accepted without a good explanation.

So how could we justify including a huge block of random symbols? The solution is quite obvious: we submit a paper that involves the running of a simple experiment, and then tell everyone that for the purposes of reproducibility, we must include a large sample of our system's random number table (in base64). We can then argue that anyone who rejects our paper is an enemy of open science.

Algorithm 4 RandoSpam

procedure RandoSpam(n)
    Run some trivial experiment that involves the use of random simulation.
    Report the results.
    Paste in n random symbols using base64.
    Make it very clear to all that rejecting the paper would make them anti-science.
end procedure

4.4 Steal Papers

Do not be deceived by the name. This next strategy is quite distinct from plagiarism.

The bigness of small negative numbers pops up all over mathematics and computer science. For example, on most 64 bit systems, if we let $x = 2^{63} - 1 = 9223372036854775807$, we have that $2x + 1 = -1$.

Furthermore, we know from pure mathematics that $\sum_{i=1}^{\infty} i = -\frac{1}{12}$.

These are deep, revolutionary ideas in the realm of paper maximization. They suggest that to get an infinitely long (or at least longer than anything seen before) paper, we just need to submit one with a small negative length.

How could we do this? It seems that it would require us to take away length from other papers, or the SIGBOVIK proceedings themselves. We claim to have already stolen space from other papers by getting this one accepted, but this is somewhat unsatisfying (the same could be said of every other paper in the proceedings). Research is ongoing and will largely depend on the security at the SIGBOVIK presentations.

5 Optimizations

Through the process of constructing these algorithms and writing them up, we discovered a couple of optimizations that should be generally applicable for the word count metric.

5.1 Itty Bitty Boppity Boo

We believe that SIGBOVIK reviewers are more concerned with a large number of pages than a large number of words. Thus, one way to massively increase the allowable word count before our acceptance probability suffers is to make liberal use of 1pt font.

5.2 Pictogram Kiloword Boosting

We are all, of course, familiar with the famous Pictogram Kiloword Equivalence Theorem (an excellent application was seen in [1]):

Theorem 2 (Pictogram Kiloword Equivalence)**.** *A picture is worth one thousand words.*

However, this simple result has some immediate corollaries that can allow word counts to be made arbitrarily large.

Corollary 1. *A picture of n words is worth $1000n$ words.*

Corollary 2 (Pictogram Paper Boosting Theorem)**.** *Let $\mathcal{P}$ be the space of all papers, and let $g : \mathcal{P} \to \mathcal{P}$ be defined such that $g(P)$ produces a paper containing a picture of P. Then if there are n words in P, the word count of $g^k(P)$ is $n \times 1000^k$.*

Thus, by successively taking pictures of our paper, we can make the word count grow exponentially while maintaining a small page count.

6 Proof of Concept

In order to make this paper the longest ever in terms of word count and information content, we combined a few of the above methods to produce the following:

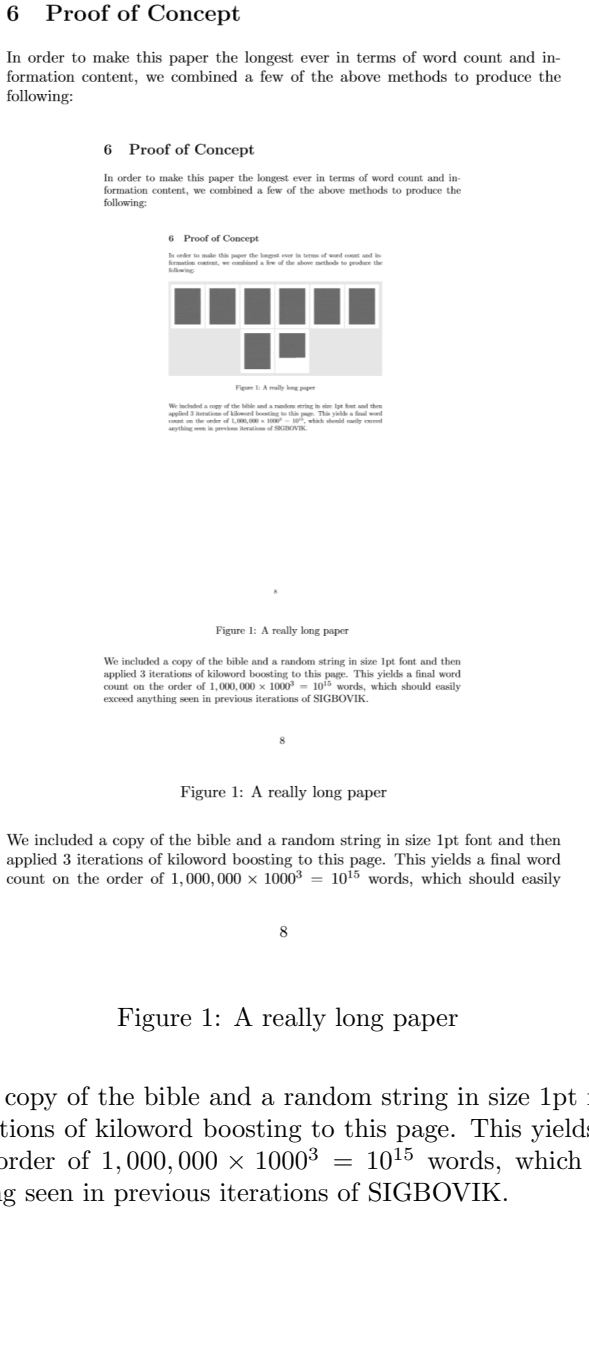

Figure 1: A really long paper

We included a copy of the bible and a random string in size 1pt font and then applied 3 iterations of kiloword boosting to this page. This yields a final word count on the order of $1,000,000 \times 1000^3 = 10^{15}$ words, which should easily exceed anything seen in previous iterations of SIGBOVIK.

References

[1] Is this the shortest sigbovik paper? In *SIGBOVIK 2018*, 2018.

[2] Thomas Bach. Is "dicong qiu. is this theshortest sigbovik paper? from 2018 sigbovik paper" the shortest sigbovik paper? In *SIGBOVIK 2019*, 2019.

[3] Mitchell Jones. Is this the tiniest sigbovik paper ever? In *SIGBOVIK 2019*, 2019.

[4] Patrick Lin. No, this is the tiniest sigbovik paper ever. In *SIGBOVIK 2019*, 2019.

[5] Dicong Qiu. Is this the shortest sigbovik paper? In *SIGBOVIK 2018*, 2018.

[6] Exasperated Reviewer. On the shortness of sigbovik papers. In *SIGBOVIK 2019*, 2019.

[7] Richard Wardin. Revisiting the shortest sigbovik paper. In *SIGBOVIK 2019*, 2019.

[8] Zach Weinersmith. In the pocket of... https://www.smbc-comics.com/comic/pocket. [Online; accessed 18-March-2021].