Technical Report: Effective Quality of Lighthouse's Greedy Approach to Attestation Packing

Satalia & Sigma Prime

November 24, 2022

Abstract

We experimentally assess the quality of the approach for solving the Attestation Aggregation and Packing Problem (AAPP, [3]) implemented in Sigma Prime's Lighthouse client. Lighthouse's approach is composed of two greedy stages (aggregation and packing) executed in sequence. The latter is based on an approximation algorithm for the weighted maximum coverage problem, and is guaranteed to find solutions non-worse than $1 - 1/e \approx 0.632$ of the optimum. However, while the theoretical results provide a lower bound on the packing quality, they give no indication of the algorithm's behaviour when applied to typical instances of the AAPP. Moreover, there is no theoretical result on the performance of the first stage (aggregation).

In order to shed some light on the algorithm's expected behaviour in the typical case, we take advantage of the exact MIP approach outlined in [4] and carry out an experimental analysis on eight days-worth of instances extracted by Sigma Prime from Ethereum's Beacon chain. The MIP approach is guaranteed to find the optimal aggregation and packing, and we can therefore benchmark Lighthouse's approach against it.

Contents

1 Introduction

In this report we outline the experimental analysis that we carried out to assess the typical quality obtained by Lighthouse's greedy approach for solving the AAPP. The cornerstone of this analysis is the exact approach described in [4] to find optimal solutions to the AAPP, which the greedy approach can be benchmarked against.

The report is organised as follows. Section 2 outlines the experimental setup. Section 3 provides a summary of the raw results, in particular with respect to the key aspects of interest: quality and run time, and some discussion of them. Finally, Section 4 outlines some possible directions for this work, to be picked up in Phase 2.

2 Experimental setup

This section aims at providing a detailed description of how the experiments were carried out. We will cover both the instances and the approach.

2.1 Instances

We have used ≈ 8 days worth of instances, each one corresponding to one slot (i.e., one block proposal) of Ethereum's Beacon chain. Table 1 illustrates some of the variability that can be observed throughout the instances. We see that most instances have a number of attesters around the 12000 mark, and a number of unique attestation data in the $[200, 400]$ range.

Percentiles	**0%**	**25%**	**50%**	**75%**	**100%**
Attesters	3461	11729	11859	12077	73999
Unique attestation data	83	205	277	372	1441

Table 1: Basic statistics on the instances used in this analysis.

Each instance comes with the corresponding greedy solution as computed by Lighthouse, however the total reward was recalculated by us.

Note. The instances reflect the relatively quiet state of the Beacon blockchain, which is not used for real transactions at the time of this writing (pre-Merge). One could expect numbers to increase as the chain gains more adoption.

2.2 Approach

We have identified an optimum for each instance by using the MIP approach described in [4]. Although the approach is discussed at length in a separate report, here are the main steps.

1. **Compute the set of candidate attestations.** This is the union of the candidate attestations for each unique attestation data. These are obtained as follows
   (a) enumerate all the attestations corresponding to maximal cliques for the graph where the vertices represent aggregated attestations, and the edges encode the disjointedness of their attester sets (this is carried out using the variant of the Bron-Kerbosch approach described in [4]) with the various mentioned optimisations,
   (b) extend the above with all the compatible unaggregated attestations,
   (c) add to the above a further aggregated attestation made up of all unaggregated attestations.
2. **Solve a maximum weighted coverage problem.** This is a problem with maximum capacity 128, and where each set corresponds to one of the above candidate attestations, and each element corresponds to an attester with a weight identical to its inclusion reward. We solve this problem using the MIP model outlined in [4].

All of the code to carry out the above steps is developed in Rust, to keep consistency with the Lighthouse codebase, and to facilitate any further development to be carried out in Phase 2. While performance wasn't a key issue for Phase 1 (beyond the practical implication of running the approach for a large number of instances), we have taken steps to avoid any unnecessary computation.

We have used the `good_lp`[1] crate for modelling and solving the weighted maximum coverage MIP model. `good_lp` provides a convenient modelling layer and bindings for many available MIP solvers (more on this later), and for these experiments we used the CBC solver [1] due to its widespread availability.

3 Results

With this experiment, we wanted to answer two main questions:

- How large is the gap between greedy and optimal solutions?
- How well does the MIP solver perform, compared to production requirements?

[1] See good_lp.

3.1 Optimality gap

Experimental results show that it exists, but, outside of certain outliers, the relative gap is very small. To illustrate the distribution of the relative gap, we provide a table of relative gap thresholds and how often they're exceeded.

Threshold	% of instances where gap > threshold
0%	47.68%
1%	0.14%
5%	0.03%

This shows that roughly 52% of the time the greedy solution was already optimal, and that in only 14 of the 51 097 instances was the gap greater than 5%.

Ultimately, the average relative gap was 0.2%, or, in absolute terms, 261 836 297 Gwei over the data collection period.

3.2 Solver performance

While the current solver isn't fit for production (as it was designed for offline use in the context of this experimental evaluation), its performance provides an indication of the computational effort required by MIP to identify optimal solutions. Results show that, for the majority of cases, the MIP approach achieves reasonable performance, albeit being slightly too slow for a production environment. This may be also down to the choice of the MIP solver backend, which is a free and open-source solver. Commercial solvers tend to exhibit better performance.

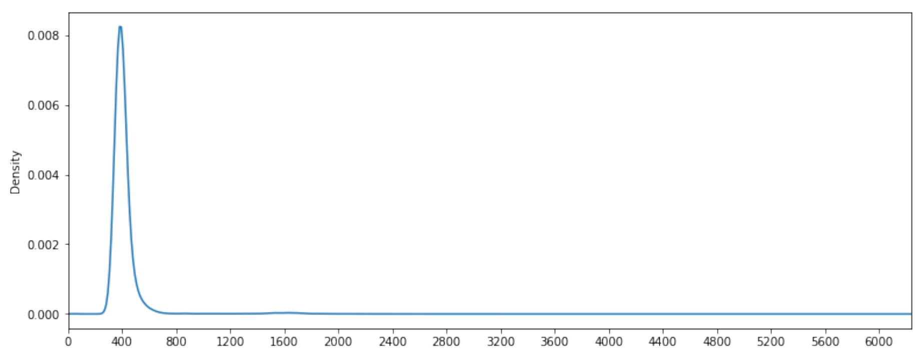

Figure 1: A density plot of solver run times.

And, to illustrate the outlier run times, a table of run time thresholds and how often they're exceeded.

Threshold	% of instances where run time > threshold
1000 ms	1.41%
2000 ms	0.14%
3000 ms	0.04%
4000 ms	0.03%
5000 ms	0.02%
6000 ms	0.00%

While performance is good most of the time, it can slow down to unacceptable speeds every once in a while. This leads us to believe that it's possible to find an optimality-reaching solution that achieves good performance.

The answers to both of our questions should be caveated with the fact that this is based on current data, on current chain activity. As time goes on, the activity on the chain, and thus these results, could change. However, these results still let us make a few observations.

3.3 Aggregating is fast, packing is slow

The approach we use for identifying the candidate attestations to include in a block is very fast (usually $\approx 15ms$ or around $3-4\%$ of the total run time). The rest ($\approx 96-97\%$) of the time is spent solving the MIP model for the weighted maximum coverage problem. This figure is at least partially explained by the choice of the MIP solver. For reference, the average run time of CBC on a standard set of MIP problems is more than 8 times larger than Gurobi's [2] (the leading commercial MIP solver). Having said that, using a commercial solver will require users to own a license, and that may not be desirable.

On the other hand, the fact that optimality-preserving candidate attestations can be computed efficiently suggests a possible improvement of Lighthouse's current heuristic, which is online and greedy and has no theoretical guarantees.

3.4 Lighthouse's approach: practice vs. theory

From the results presented in the previous section, it is clear that the greedy approach finds the optimum quite often (at least on this particular data set) and, when it doesn't, it doesn't typically get very far from it. There are a few pathological cases where the gap is much larger (e.g., $\approx 40\%$ off the optimum) but these are not common, and at this stage we cannot assess if it is due to Lighthouse's greedy aggregation stage, or its greedy packing stage.

At any rate, when factoring in both solution quality and run time, the approach seems to strike a reasonable balance that yields profitable proposals most of the time.

Qualifying the (financial) loss in profit for the cases where Lighthouse's approach doesn't reach the optimum is beyond the scope of this experimental evaluation. However, from an optimisation perspective, it may make sense to investigate methods that are either exact and sufficiently fast or that, while not being exact, can deliver better performance than the current approach more frequently.

3.5 Performance of the MIP solver

While we were not aiming for real-time performance in this project, we were positively surprised by the performance of the MIP approach, which allowed to find optimal solutions for the packing part of the problem in less than $0.5s$ more than 93% of the time.

We have some intuition around this. Here are what we think may be the main contributors.

- **Scale**. The number of *decision variables* (i.e., the ones modelling whether a given attestation is added to the block or not) is rather low, often within a few thousands. This is only possible because the job of restricting the search space to the cliques that are maximal with respect to attester coverage has been dealt with at the previous step.

- **Additive contribution**. Each distinct attestation data comes with a handful of aggregated attestations which overlap in terms of attester coverage, i.e., including one of them in the solution affects the contribution of including the others. This is not the case for attestations with different attestation data, which are the majority. For these, the contribution is additive, and choosing one doesn't affect the value of choosing another. Depending on its implementation, a solver may be able to use this information to obtain efficient bounds on the maximum reward that can be achieved when extending a solution with an attestation. Calculating tight bounds is one of the key strategies MIP solvers use to avoid searching non-promising parts of the search space.

Even if the performance of the MIP approach is better than we expected, there are two main challenges to its inclusion in Lighthouse.

1. The performance may not be enough for the real-time scenario, where it matters how quickly a validator can propose a new block. In particular, the run time of the MIP approach varies quite wildly from $\approx 200ms$ to more than $6s$.

2. The MIP solver would represent a dependency, which would have to be audited properly in order to be trusted[2], in this sense it may be better to use an approach that can be fully included in Lighthouse's codebase and audited accordingly.

In Section 4 we discuss some possible alternatives to the MIP approach.

3.6 Predicting metrics of interest

We have tracked some of the performance indicators of interest, namely

[2] Note that, while the first point could be mitigated with the decomposition method that we proposed in [4], if said method relies on the MIP approach, a MIP solver would still be a dependency.

- time taken by the MIP to find an optimal solution,
- optimality gap of the greedy solution, i.e., the relative difference in quality between the quality of the greedy solution and the quality of the optimal solution,

and explored their relationship with measurable features (both of the instance and of the output of the aggregation phase).

MIP run time. It appears that there is a relatively strong linear correlation between the number of attesters, the number of unique attestation data, and the performance of the MIP solver. A simple linear model can predict the time taken by the MIP solver to find a solution with a reasonable accuracy ($R^2 \approx 0.84$). Let a be the number of attesters in the instance, and d the number of unique attestation data, then the expected time (in milliseconds) p to solve the MIP model is

$$p = -597.85 + a \times 0.085 + d \times -0.093. \tag{1}$$

This is something that can be used to estimate, for instance, how this particular MIP approach would scale as the size of the instances change. We imagine that both a and d would increase as the Beacon blockchain is adopted, which would drive up the expected run time of the MIP approach.

Optimality gap. Unfortunately, we haven't found any clear correlation between measurable features and the optimality gap. It would be interesting to understand what makes a problem hard to solve for the greedy approach, but at this stage there isn't a clear feature explaining this metric.

4 Future work

We have outlined a few areas for future research, specifically around designing algorithms that can be used in production and that would perform better than the current heuristics. Broadly speaking, these approaches are somewhat in-between the exact MIP approach and the greedy approach implemented by Lighthouse (see blue area in Figure 2). The rest of this section will outline some more specific directions.

4.1 Decomposition approach

Due to the fact that the MIP approach was fit for purpose for this experimental evaluation, we didn't implement the decomposition approach outlined in [4]. It is our belief that the decomposition approach would perform significantly better than the MIP approach. Our belief is based on the following observations

1. solving the weighted maximum coverage problem represents the highest time spender in the MIP approach,

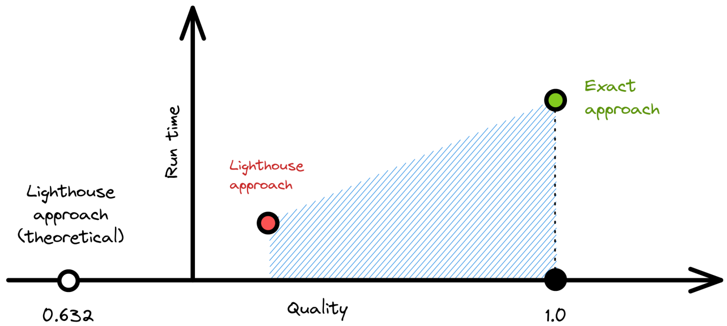

Figure 2: Spaces of approaches to be explored.

2. the weighted maximum coverage problem is NP-hard, therefore solving large problem is much harder than solving small problems, and
3. the decomposition approach involves solving much smaller weighted maximum coverage problems.

We therefore believe that this is a meaningful line of research.

4.2 Replacing the MIP approach

So far we have solved the weighted maximum coverage problem using a MIP solver. Because of the limitations outlined previously around using a MIP within Lighthouse, we have considered some options to replace the MIP.

The one that currently seems most promising, but only applicable in the context of a decomposition approach, is a custom-built branch & bound tree search. There are several bounds that we can use to speed-up this type of search, and the added complexity to the Lighthouse codebase would be limited. Moreover, it wouldn't require any additional library, removing the uncomfortable dependency on third-party IP. We have outlined such a procedure in our previous report.

4.3 Pruning candidate attestations

There are some strategies that can be used to further reduce the size of the weighted maximum coverage problems to be solved, e.g., pruning attestations that are dominated by others with respect to attester coverage. These strategies require additional computations, and we cannot exclude that the additional computation could outweigh the advantage of dealing with a smaller problem, therefore an appropriate trade-off between pre-processing and solving must be identified (possibly experimentally).

4.4 Other heuristic approaches

Exact approaches are not the only possibility when it comes to algorithms that would be fit for production. Depending on the *definition of good*, we may want to consider heuristics that don't guarantee optimality, but that perform better than the current approach. These could even combine some of the components already outlined in the previous report. Some examples are

1. greedy approach → local search,
2. maximal clique enumeration → greedy approach,
3. maximal clique enumeration → greedy approach → local search,
4. maximal clique enumeration → branch & bound.

These approaches need to be compared experimentally, and the corresponding quality *vs.* complexity trade-off needs to be be evaluated.

Acknowledgements

We thank Sigma Prime for their support in providing the details of the AAPP, as well as a sample of instances from Ethereum's Beacon chain.

References

[1] John Forrest and Robin Lougee-Heimer. Cbc user guide. In *Emerging theory, methods, and applications*, pages 257–277. INFORMS, 2005.

[2] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022.

[3] Satalia Team. Technical report: Attestation packing problem (abridged). Technical report, Satalia (NPComplete Ltd), 2022.

[4] Satalia Team. Technical report: Exact approaches for the attestation aggregation and packing problem. Technical report, Satalia (NPComplete Ltd), 2022.