Exact Approaches for the Attestation Aggregation and Packing Problem

Satalia & Sigma Prime

May 19, 2022

Abstract

We explore two exact approaches for solving the Attestation Aggregation and Packing Problem (AAPP, [6]). Both approaches are optimality-preserving.

The first approach (which we refer to as the *MIP approach*) involves a first stage to deal with the *aggregation* part of the problem, and a second stage, based on a a mixed integer programming (MIP) formulation, to deal with the *packing* part of the problem. The second approach (which we refer to as the *decomposition approach*) decomposes the problem into a main problem and many sub-problems. The main problem can be solved efficiently using dynamic programming, the sub-problems are are smaller instances of a special case of the AAPP, which can be solved optimally with any exact approach.

The purpose of the decomposition approach is to help with the potential scalability issues of the MIP approach on larger instances.

Contents

1 Introduction **2**

2 The MIP approach **2**
2.1 Observations 2
2.2 Aggregation problem 3
2.3 Packing problem 7

3 Decomposition approach **8**
3.1 Sub-problems 9
3.2 Main problem 10

1 Introduction

The present document outlines two exact approaches for solving the Attestation Aggregation and Packing Problem (AAPP) defined in [6]. For convenience, we will not include the formal definition of the problem here. Instead, we will assume familiarity with [6] and commit to use the same notation. Where a symbol is used in different contexts, e.g., the symbol e for epochs and set elements, the meaning will be clear from the context.

While the main use case for the proposed exact approaches is to find optimal solutions to the AAPP, we will take scalability into account for practical reasons.

2 The MIP approach

The MIP approach is based on the idea of re-framing the AAPP as a combination of an *aggregation problem* and a *packing problem*. This is similar in spirit to the approach taken by Sigma Prime's Lighthouse[1]. Like Lighthouse, the MIP approach solves these two problems in successive stages. Unlike Lighthouse, the MIP approach is complete and is guaranteed to (eventually) find an optimal solution.

2.1 Observations

Let $A^\dagger \subseteq 2^A$ be the set of sets of attestations satisfying the aggregation conditions, i.e.,

$$A^\dagger = \{B \mid V_a \cap V_b = \emptyset \wedge d_a = d_b,\ \forall a, b \in B,\ B \in 2^A\} \tag{1}$$

Note that any set $S \subseteq A^\dagger, |S| \leq N$ is a feasible solution of the AAPP, and that each element of $A^\dagger$ represents a valid[2] aggregated attestation. The re-framing of the AAPP behind the MIP approach is based on the following two Observations

O1 Because every attestation, regardless of the size of its attester set, takes up exactly one of N slots within a block, including an attestation $a \in A^\dagger$ with attester set V_a in a solution is non-worse than including an attestation $b \in A^\dagger$ with attester set $V_b \subseteq V_a$,

O2 An attester $v \in V$ appearing in a solution S contributes only once to the reward of that solution, regardless of how many times it appears in the attester sets of attestations included in S.

The consequence of *O1* is that attestations that are *maximal* with respect to attester coverage are the best (or at least non-worse) candidates for inclusion in a solution of the AAPP. The set of such attestations is the set

$$A^\star = \{B \mid \nexists C \in A^\dagger, V_B \subset V_C, B \in A^\dagger\} \tag{2}$$

[1] Ethereum consensus client.

[2] In the sense of the conditions for aggregation.

In other words, all attestations in $A^\star$ are *non-dominated* with respect to the coverage of their attester set, i.e., they are non-worse than any other attestations in $A^\dagger$ at covering their particular attester set. Note that these attestations are also *maximally aggregated*, i.e., they cannot be aggregated with any other attestation in $A^\dagger$. If they were, they wouldn't be maximal with respect to attester coverage[3]. Note that, according to the above definition, $A^\star$ could contain aggregated attestations that are equivalent with respect to attester coverage. This is not a limitation for our approach.

This has direct implications for the search of an **optimal** solution because, given a set of attestations A, there is at least some optimal solution S to the AAPP which is a subset of $A^\star$. It has also implications for the search of a **good quality** (but not necessarily optimal) solutions since, due to the fixed capacity N, attestations not in $A^\star$ are less likely to be part of a good-quality solution.

Considering $2^{A^\star}$, as opposed to $2^{A^\dagger}$, as the search space has two main advantages

- first, the former represents a smaller search space, i.e., $|2^{A^\star}| \leq |2^{A^\dagger}|$, and
- second, because every element of $A^\star$ already satisfies all the aggregation constraints, we can safely ignore them going forward.

Computing $A^\star$ is what we refer to as the *aggregation problem*, which is the focus of the first stage of our MIP approach.

The consequence of *O2* is that, once the aggregation problem has been solved, the AAPP reduces to choosing N aggregated attestations from $A^\star$ so as to maximise the reward to be gained by including the votes of the attesters they cover. This is a *weighted maximum coverage problem*, which needs to be solved exactly in order for our approach to guarantee optimality. This is what we refer to as the *packing problem*.

2.2 Aggregation problem

In this section we outline a method to compute $A^\star$ using a graph representation of the set of attestations A.

Let $G = (V, E)$ be an undirected graph with vertices V and edges E, where $V = A$, and

$$E = \{(a, b) \mid \{a, b\} \in A^\dagger\} \tag{3}$$

i.e., each vertex represents one of the attestations in A, and each edge encodes the compatibility for aggregation of the attestations it connects.

A *clique* of G is a set $C \subseteq V$ such that each pair of vertices in the clique is connected by an edge, i.e., $(a, b) \in E, \forall a, b \in C, a \neq b$. Note that, due to our encoding of G, a clique represents an aggregate attestation from attestations in A, and the set of all such attestations is $A^\dagger$. A *maximal clique* is a clique

[3] If they were, it would be possible to find an attestation with the same attestation data and disjoint set of validators to aggregate them with, which would lead to a higher attester coverage, which directly contraddicts the definition.

that cannot be further extended with another vertex, and thus corresponds to a maximally aggregated attestation from A, and the set of all such attestations is $\tilde{A} \subseteq A^\dagger$. Note that while the elements of $\tilde{A}$ are maximally aggregated attestations, they are not necessarily non-dominated, and therefore $\tilde{A} \supseteq A^\star$. For instance, consider the two hypothetical maximally aggregated attestations $a, b \in \tilde{A}$, their respective attester sets could be

$$V_a = \{v_1, v_2, v_3\} \tag{4}$$

$$V_b = \{v_1, v_2, v_3, v_4\}. \tag{5}$$

Obviously a is not maximal with respect to attester coverage, since choosing b is non-worse than choosing a in terms of covering $\{v_1, v_2, v_3\}$. In other words, $a \notin A^\star$ but $b \in A^\star$.

In the following, we present a method to compute $A^\star$ by first computing $\tilde{A}$ by enumerating all the maximal cliques, and then eliminating all the aggregated attestations that are dominated with respect to attester coverage, i.e., those whose attester sets are proper subsets of of the attester set of some other attestation in $\tilde{A}$. Again, this is not a strict requirement for the approach, but is likely to make the search space more compact.

Note. For the purpose of solving the AAPP with the MIP approach, it would be sufficient to choose N attestations from $A^\dagger$. Computing $A^\star$ is a way to reduce the search space of the packing problem while preserving the optimality of the approach.

Most of the existing algorithm for enumerating maximal cliques are variants of the Bron-Kerbosch [1] algorithm. This is a recursive algorithm that maintains three sets of vertices R, P, and X respectively modeling the maximal clique being built, the vertices that could be included in the clique, and the vertices that won't be included in the clique. The essence of the Bron-Kerbosch algorithm is summarised in the pseudo-code below, where we denote by $N(v)$ the set of vertices connected by some edge to a vertex $v \in V$ of a given graph $G = (V, E)$.

Algorithm 1 Original Bron-Kerbosch

```
procedure BronKerbosch(R, P, X)
    if P ∪ X = ∅ then
        report R as maximal clique
    end if
    for v ∈ P do
        BronKerbosch(R ∪ {v}, P ∩ N(v), X ∩ N(v))
        P ← P \ {v}
        X ← X ∪ {v}
    end for
end procedure
```

While the original Bron-Kerbosch algorithm would correctly produce the set of maximal cliques in a graph, the algorithm has been improved since its first inception in 1973. Two main developments are the use of *pivoting* [4, 2] and the use of *vertex ordering* [3]. The above techniques can be applied to the general maximal clique enumeration problem and are therefore relevant for us.

Pivoting. The idea of pivoting is based on the observation that, given a vertex $p \in P \cup X$, any maximal clique must contain p or one of its non-neighbors, i.e., $P \setminus N(p)$ (otherwise, the clique could be extended by adding p to it), therefore only non-neighbors of p need to be explored while extending R (otherwise a non-maximal clique is being explored). A strategy for choosing p that has proven to be effective both theoretically and experimentally is to choose

$$p = \arg\min_{p \in P \cup X} |P \setminus N(p)|. \tag{6}$$

This variant would roughly translate to the following pseudo-code.

Algorithm 2 Bron-Kerbosch with pivoting

```
procedure BronKerbosch(R, P, X)
    if P ∪ X = ∅ then
        report R as maximal clique
    end if
    p ← arg min_{p∈P∪X} |P \ N(p)|
    for v ∈ P \ N(p) do
        BronKerbosch(R ∪ {v}, P ∩ N(v), X ∩ N(v))
        P ← P \ {v}
        X ← X ∪ {v}
    end for
end procedure
```

Vertex ordering. The vertex ordering variant is based on the idea that, by choosing the ordering in which the recursive calls are made, i.e., by choosing the ordering in which vertices in P are explored, one can reduce the size of the recursive search tree. As for the possible orderings, using a *degeneracy ordering* has been associated with better worst-case run time guarantees. In order to obtain a degeneracy ordering, one approach is to start with an empty ordering and iteratively remove a minimum degree vertex from the graph and append it to the ordering until the graph is empty. This is done in the outermost level of the recursion, while the innermost levels still use a pivoting strategy.

Algorithm 3 Bron-Kerbosch with ordering

```
procedure BronKerboschOrdering(G = (V, E))
    v_1, v_2, ..., v_n ← ComputeOrdering(G)
    for i ∈ {1, ..., n} do
        P ← {v_j | j > i} ∩ N(v_i)
        R ← {v_i}
        X ← {v_j | j < i} ∩ N(v_i)
        BronKerbosch(R, P, X)
    end for
end procedure
```

Aside from the above improvements to the original Bron-Kerbosch algorithm, the fact that $V = A$ allows us to consider two additional optimisations. First, some of the attestations in A will expectedly be unaggregated, and will therefore correspond to vertices connected with all the vertices for attestations that do not include them, which could be many. This can increase the cost of enumerating all cliques, and can be handled as a pre-processing and a post-processing step, discussed below under *unaggregated attestations*. Second, the aggregation conditions state that two vertices cannot be connected if their attestation data differs. This means that $G = (V, E)$ as defined above could contain many disconnected sub-graphs partitioned by attestation data and, as a consequence, the set of maximal cliques will be partitioned in the same way. Solving the maximal clique enumeration for each attestation data independently reduces the size of the graphs to be processed, which we cover *attestation data partitioning* below.

Unaggregated attestations. We want to reduce the complexity of enumerating all the cliques for graphs arising in the context of the AAPP. Such graphs are of the form $G = (V = A, E)$. In particular, we're interested in dealing with unaggregated attestations, which are compatible with many cliques and can therefore contribute to generating a large set of maximal cliques.

Let us denote by $A^1 = \{a \mid a \in A,\ |V_a| = 1\}$ the set of unaggregated attestations in A. Our strategy for dealing with these is to remove them from G altogether, enumerate all the maximal cliques for $G' = (A \setminus A^1, E')$, where $E' = \{(a, b) \mid (a, b) \in E, a, b \in A \setminus A^1\}$, and then add all $a \in A^1$ back in all the compatible cliques. We denote this set as $\tilde{A}'$.

Note that in general $\tilde{A}' \subseteq \tilde{A}$. This is not a limitation, as it can be shown that $\tilde{A}' \supseteq A^\star$ holds. In other words, the additional cliques that would have been generated by considering A^1, i.e., $\tilde{A} \setminus \tilde{A}'$ are necessarily dominated by the ones in $\tilde{A}'$.

(Potential) attestation data partitioning. As mentioned above, the graph $G = (V, E)$ is partitioned by attestation data. This suggests that the maximal clique enumeration can be decomposed, which may have a positive impact on the performance. Let $A_d = \{a \mid a \in A,\ d_a = d\},\ \forall d \in D$. We can then generate all

the aggregated attestations that are maximal with respect to attester coverage $A_d^\star$ for A_d, by defining $G_d = (A_d, E)$ where E is defined in the obvious way. And following the approach described above. Then, the set of all maximal cliques for A can be then defined as

$$A^\star = \bigcup_{d \in D} A_d^\star. \tag{7}$$

The intuition is that calculating $A_d^\star$, $\forall d \in D$ is more efficient than computing $A^\star$ for A as a whole[4].

Other approaches. The presented algorithms are well-established, and tend to achieve good performance on generic graphs, and have the advantage of simplicity, which can be a desirable property for the type of use case discussed here. However more sophisticated algorithms exist. Most of these target graphs with particular properties. On these graphs, these algorithms tend to provide better worst-case complexity guarantees. Some others address generic graphs, and can provide better guarantees than the algorithms based on Bron-Kerbosch. We take stock of these algorithms, which can be explored in a successive phase if necessary.

Note. whichever technique is chosen to enumerate the set $A^\star$, its efficiency is crucial to the viability of both approaches proposed here. In the following, we assume that $A^\star$ has been found in a way or another, and is available as an input to the packing stage.

2.3 Packing problem

As we have mentioned above, the set of all aggregated attestations that are maximal with respect to attester coverage $A^\star$ is an optimality-preserving subset of the search space of the AAPP. Based on this premise, the AAPP can be easily formulated as a *weighted maximum coverage problem.*

The weighted maximum coverage problem involves choosing at most k from a set of sets $S = \{S_1, \ldots, S_n\}$, where each element $e \in \bigcup_{i \in \{1,\ldots,n\}} S_i$ is associated with a weight $w(e) \in \mathbb{N}^{\geq 0}$. The goal of the problem is to maximise the sum of the weights of the elements that are part of at least one of the k sets that are chosen to be part of the solution. Like its non-weighted variant, the weighted maximum coverage problem is unfortunately NP-hard. It be formulated as a

[4]This needs to be proven experimentally.

mixed integer program (MIP) as follows

$$\textbf{maximise} \sum_{e \in E} w(e_j) \cdot y_j \tag{8}$$

$$s.\ t. \sum x_i \leq k \tag{9}$$

$$\sum_{e_j \in S_i} x_i \geq y_j$$

$$y_j \in \{0, 1\}$$

$$x_i \in \{0, 1\}$$

where $S = \{S_1, \ldots, S_n\}$ is the set of sets that can be included in a solution and $E = \{e_1, \ldots, e_m\} = \bigcup_{S_i \in S} S_i$ is the set of elements in any of the sets. The variables $x_i \in \{1, \ldots, n\}$ and $y_j \in \{1, \ldots, m\}$ encode, respectively, the decision to choose a set S_i in to be part of the solution, and the fact that element e_j is covered by the solution.

Mapping. Under the premise that $A^\star$ is available, the mapping of the AAPP to this problem is trivial. Let $S = \{V_a \mid a \in A^\star\}$, and $k = N$, where N has the meaning defined in [6]. A solution for the weighted maximum coverage problem defined this way is a solution to the original AAPP. Moreover, an optimal solution to this problem is an optimal solution of the original AAPP.

This formulation can be directly plugged into a MIP solver to find an optimal packing of N maximally aggregated attestations.

Other exact approaches. Of course, the weighted maximum coverage problem can be solved optimally by any other exact approach. Such an approach, e.g., constraint programming (CP), would likely benefit from a different modeling. In this section we focus on MIP because of the wide availability of these kinds of solvers.

Greedy algorithm. A greedy algorithm exists and is indeed the algorithm currently employed by the Lighthouse client developed by Sigma Prime. This greedy algorithm has a guaranteed approximation ration of $1 - 1/e \approx 0.623$ of the optimum, which is not suitable for the purpose of this work, but may be of interest for the follow-up implementation of an efficient algorithm. Note that in Lighthouse, the packing problem is defined on a set of attestations that have been pre-aggregated heuristically, which differs from the approach presented here.

3 Decomposition approach

Solving the AAPP using a MIP solver may represent a suitable approach, unless the problem is too large. For these scenarios, we propose to use a decomposition approach.

The main idea behind this approach is to reduce the complexity of solving the AAPP by separating the parts that can be handled efficiently with *dynamic programming* from the ones that are strictly *NP-hard*. In particular, this approach is based on the observation that, when choosing which attestations to include in a solution, the contribution of attestations with different attestation data is additive, i.e.,

$$d_a \neq d_b \Rightarrow Reward\{a, b\} = R(\{a\}) + R(\{b\}),\ \forall a, b \in A. \tag{10}$$

This suggests that, by partitioning an instance of the AAPP by attestation data, we could re-frame it as the problem of choosing the number $q_d \in \mathbb{N}^{\geq 0}$ of sets of attestations to include from each A_d, where $d \in D$. This is reminiscent of the Knapsack Problem [5], which can be solved efficiently with dynamic programming.

In order for a solution to the AAPP to be optimal, one needs to choose the *best* q_d attestations for $d \in D$, i.e., the ones that will collectively provide maximal reward. This includes of course both attestations in A_d but also aggregated attestations $A_d^\dagger$ from attestations in A_d.

This is, once again a *weighted maximum coverage problem* (albeit expectedly a much smaller one), and thus NP-hard. The value of this decomposition is that it allows us to treat the full AAPP as a combination of a dynamic programming problem (which we will refer to as **main problem**) and many small NP-hard problems (which we will refer to as **sub-problems**), as opposed to one large NP-hard problem. This is typically a much better outlook.

In the rest of this section, we will characterise both the main problem and the sub-problems. Because the former depends on the latter, we will discuss the sub-problems first.

3.1 Sub-problems

Let $d \in D$ be a unique attestation data in the input. We denote by A_d the set of all attestations for d, and $k \in \mathbb{N}^{\geq 0}$ a positive integer. We now consider the AAPP problem defined by $A = A_d$ and $N = k$, and denote by $g(d, k)$ any optimal solution for such AAPP. For a given $d \in D$ and $k \in \mathbb{N}^{\geq 0}$, $g(d, k)$ can be found using the MIP approach presented in Section 2, or any equivalent exact approach.

Note For some $k \in \mathbb{N}^{\geq 0}$ it is possible that $|g(d, k)| < k$. This represents a sort of *fixpoint* for g, and reflects two possible scenarios, either

- there are fewer than k sets of attestations from 2^{A_d} to choose from, or
- $R(g(d, k)) = R(g(d, k-1))$, i.e., adding more sets of attestations doesn't increase the value of $g(d, k)$, i.e., $V_{g(d,k-1)} \supseteq \{v \mid v \in V_{A_d},\ r(e(d), v) > 0\}$.

3.2 Main problem

Now that we have defined the sub-problems and that we can refer to their optimal solutions, we have all the ingredients to define the main problem as a dynamic programming problem.

Let $D = \{d_1, \ldots, d_n\}$ be a set of $n \in \mathbb{N}^{\geq 0}$ distinct attestation data, and let $m \in \mathbb{N}^{\geq 0}$ be a non-negative integer. We observe that, under the premise that $g(d_i, k)$ for some $i \in \{1, \ldots, n\}$ and some $k \in \{0, \ldots, m\}$ is the optimal (most rewarding) set of k aggregated attestations $2^{A^{\dagger}_d}$, the AAPP reduces to finding an appropriate value $q_d \in \{0, \ldots, m\}$ for every d such that

$$\sum_{d \in D} q_d \leq m, \tag{11}$$

in other words, $q_{d_1}, \ldots, q_{d_n}$ uniquely identify a packing S of at most m aggregated attestations, where for each $d \in D$ we're choosing the q_d best attestations to include. The actual packing can then be derived as

$$S = \bigcup_{d \in D} g(d, q_d). \tag{12}$$

To find the optimal solution, we proceed with a dynamic programming approach. We denote by $f(k, m)$ the *optimal* packing of m aggregated attestations $\bigcup_{i \in 1, \ldots, k} 2^{A^{\dagger}_{d_i}}$ from the attestation sets of the first k attestation data. As customary, we first identify a base case, and then define the non-base cases recursively.

Base case. Note that

$$f(0, m) = \emptyset, \quad \forall m \in \mathbb{N}^{\geq 0} \tag{13}$$

$$R\left(f(0, m)\right) = 0 \tag{14}$$

i.e., the only solution for the main problem considering no attestation data at all[5] is the empty set, which yields a reward of 0, no matter the available capacity.

Non-base cases. Based on this, we can build our non-base case as follows

$$f(k, m) = \underset{q \in \{0, \ldots, \min(m, |g(k,m)|)\}}{\arg\max} R\left(f(k-1, m-q) \cup g(d_k, q)\right) \tag{15}$$

i.e., the best packing $f(k, m)$ of at most m sets of aggregatable attestations considering only the first k attestation data $\{d_1, \ldots, d_k\} \subseteq D$ is the set that maximises the total reward that can be obtained by choosing the best $m - q$ aggregated attestations from either of the $f(k-1, m-q)$, and the best q aggregated attestations from $g(d_k, q)$.

[5] It is possible to define the base case as $f(1, m)$ however its definition is already encompassed by the definition for the non-base cases, so starting from 0 is more succinct.

Note that the domain $\{0, \ldots, \min(m, |g(k,m)|)\}$ of q takes into account the fact that it is possible that $g(k,m) < m$ for some values of m, and therefore adding more sets than available in $g(k,m)$ has no meaning.

The optimality of this approach is guaranteed by the fact that $f(k-1, m-q)$ is the optimal packing for attestation data $\{d_1, \ldots, d_{k-1}\}$ and a maximum capacity of $m-q$ and, when extending our options to attestation sets for d_k, we consider all values of q from 0 (which corresponds to not including any attestation set for d_k) to m (which corresponds to choosing all m sets from the best m attestation sets for d_k). Note that at any time, only one value of $q_i,\ i \in \{1, \ldots, n\}$ is chosen. The fact that attestation sets for different attestation data do not overlap in terms of their contribution to a solution, means that the reward of a given packing is additive, and can be computed as

$$R(f(k,m)) = \max_{q \in \{0,\ldots,m\}} R(f(k-1, m-q)) + R(g(d_i, q)). \tag{16}$$

It follows that $f(n, N)$, i.e., the optimal packing considering all attestation data and a maximum capacity of N (where N has the same meaning as in [6]), corresponds to the optimal solution of the original AAPP.

Note that the property $|f(k,m)| \leq m$ in Equation 11 implies that it is possible that $|f(k,m)| < m$ for some $m \in \mathbb{N}^{\geq 0}$. Like for $g(\cdot,\cdot)$, this suggests a sensible stopping condition for the search for $f(k,m)$. This stopping condition can be used to avoid exploring $f(k, m+1)$ if $|f(k,m)| = |f(k,m-1)|$.

Improvements. Given that finding each $g(d,k)$ is NP-hard, it makes sense to help the MIP approach solve the sub-problems by injecting as much information as we can from the main problem. One information that we have available when requesting a $g(d,k)$ is the reward of $g(d,k-1)$, i.e., $Reward(g(d,k-1))$. We know that adding more aggregated attestations to $g(d,k)$ must at least achieve the same reward, plus the reward to be obtained by greedily including the single most rewarding aggregated attestation not already in $g(d,k-1)$. Let denote such attestation by $g^+_{d,k-1}$. We can now inject the following additional constraint in the MIP model of Equation 9

$$\sum_{e \in E} w(e_j) \cdot y_j \geq R(g(d,k-1)) + R\left(g^+_{d,k-1}\right). \tag{17}$$

Our expectation is that such a constraint could help prune branches of the MIP search tree that cannot reach a better-than-greedy quality.

References

[1] Coen Bron and Joep Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, sep 1973.

[2] Frédéric Cazals and Chinmay Karande. An algorithm for reporting maximal c-cliques. *Theoretical Computer Science*, 349(3):484–490, 2005.

[3] David Eppstein and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. In *International Symposium on Experimental Algorithms*, pages 364–375. Springer, 2011.

[4] Ina Koch. Enumerating all connected maximal common subgraphs in two graphs. *Theoretical Computer Science*, 250(1):1–30, 2001.

[5] Silvano Martello and Paolo Toth. Algorithms for knapsack problems. *North-Holland Mathematics Studies*, 132:213–257, 1987.

[6] Satalia Team. The attestation packing problem (abridged). Technical report, Satalia (NPComplete Ltd), 2022.