tags: eth2

algorithms

H!

Context:

I work at Prysmatic Labs, and at the moment looking for ways to improve how attestations are aggregated in Prysm client. The problem is notoriously hard (as in NP-Complete hard), so I wanted to ask whether anyone has any suggestions on top of what I've already found.

Below is the self-contained abridged version of what I am working on, full version is also available. Please, let me know if there're some approaches I've missed or if some of the explored approaches have more to them than I've noticed.

Any help in this exploratory analysis is highly appreciated!

Background

In order to increase profitability, attestations must be aggregated in a way to cover as many individual

attestors as possible. There is an important invariant: overlapping attestations can not be merged/aggregated.

// Sample attestations: A0 = [[0 0 0 1 0], // A [0 0 1 0 0], // B [0 1 0 1 0] // C]

// List can be transformed into: A1 = [[0 0 1 1 0], // A + B, most profitable - will be used for BLS aggregation [0 1 0 1 0] // C]

// Or, even better: A2 = [[0 0 0 1 0], // A [0 1 1 1 0] // B + C, most profitable - will be used for BLS aggregation]

The main objective is to find an approximation algorithm that will result in a solution which is as close to the optimal one as possible

.

Algorithms are analyzed using \alpha

-approximation method, where 0 \le \alpha \le 1

is the approximation ratio, and an algorithm with a given value of \alpha

produces solution which is at least \alpha

times the optimal

value.

In our problem, solutions are scored by their cardinality: the more participants we have within a single aggregated item the better, with the maximum possible size equal to all the validators in a given committee.

Formal Problem Statement

Definition

Def (Attestation Aggregation):

AA(U, S, k) \to S'

Let U

be a finite set of objects, where |U| = n

. Furthermore, let S = {S_1, ..., S_m | S_i \subseteq 2^U}

be a collection of its subsets, where \bigcup_{i = 1}^m S_i = U

i.e. all u \in U

are present in one of the elements of S

. Then, Attestation Aggregation

(AA) is the problem of finding S' \subseteq S

that covers at least k \in [1..n]

elements from U

, and sets in S'

are disjoint:

|\bigcup\limits_{T \in S'}T| \ge k

and

S'_i \cap S'_j = \emptyset, \forall i, j \in [1..m], i \ne j

Ideally, we want \bigcup\limits_{T \in S'}T

to have maximum cardinality, that's k = |U|

i.e. all u \in U

are covered by S'

:

|\bigcup_{T \in S'}T| = |U|

Since BLS doesn't allow merging overlapping signatures, there's that additional constraint of making sure

that all elements of S'

are pairwise disjoint.

To summarize: given a family of sets S

, we need to find a subfamily of disjoint sets S'

, which have the same (or close to same) union as the original family.

The problem is NP-Complete and only allows for logarithmic-factor polynomial-time approximation.

Comparison to Known Problems

Attestation Aggregation (AA) vs Minimum Set Cover (SC)

In the MSC we have the very same input set system (U, S)

, but our target S'

is a bit different: we want to find a full cover of U

with minimal |S'|

.

With AA, partial (if still maximal) coverage is enough, there's no constrains on cardinality of S'

, and all elements of S'

are pairwise disjoint.

Attestation Aggregation (AA) vs Exact Cover (EC)

Again, we start from the same set system (U, S)

, and the EC matches the ideal case of our problem when there exists an optimal solution within a given input S

. So, if input list of attestations form (by itself or as any combination of its subsets) a full partition of U

, the resultant S'

for both EC and AA coincide.

There is on important difference in AA: it allows for partial covers.

Attestation Aggregation (AA) vs Maximum Coverage (MC)

In the MC problem, we want to find up to k

subsets that cover U

maximally:

|S'| \le k \land \mathop{argmax}\limits_{S'} |\bigcup\limits_{T \in S'}T|

Important thing to note is that in its conventional form MC doesn't require elements of S'

to be disjoint, which is a problem for our case – as overlapping attestations cannot be aggregated.

So, the important differences of AA include: no constraints on cardinality of S'

, requirement of pairwise disjoint elements in S'

.

MC can still be utilized for our purposes: since there exists an approximation algorithm with \alpha \approx 0.6

(pretty impressive) we can rely on it to build partial solution by gradually increasing k

(see the Possible Solutions section below).

Possible Solutions

So, our problem is closely related to set cover kind of problems to which there exist several possible approaches, none of which enjoys having a deterministically optimal solution.

Several closely related NP/NP-hard problems (and their variants) have been considered:

- Set Cover Problem
- Exact Cover
- Maximum Coverage Problem
- Set Packing
- Max Disjoint Set

Set Cover

The Set Cover problem is one of Karp's 21 NP-Complete problems.

It seems natural to start from the base covering problem because it serves as a foundation for other

problems, it has a greedy algorithm solver with ln(n)

approximation to optimal, and with some effort

we can even make that greedy solver run in a linear time!

Def (Minimum Set Cover):

MSC(U, S) \to S'

Let U

be a finite set of objects, where |U| = n

. Furthermore, let S = {S_1, ..., S_m | S_i \subseteq 2^U}

be a collection of its subsets, where \bigcup_{i = 1}^m S_i = U

. Then, Minimum Set Cover

(MSC) is the problem of covering U

with a subset S' \subseteq S

s.t |S'|

is minimal.

Framed like that, this problem doesn't abstract attestation aggregation completely. While MCS produces

a cover of U

, S'

may contain subsets with overlapping elements from U

, and as such can't be used as input to aggregation function. So, we need to add an extra constraint – making sure that all elements in S'

are pairwise disjoint.

Def (Minimum Membership Set Cover):

MMSC(U, S, k) \to S'

The same set system as in MSC, with additional requirement on how many times each u \in U

can occur in elements of S'

i.e. \mathop{max}\limits_{u \in U} |{T \in S'| u \in T}| \le k

, for a nonnegative k \in [1..m]

.

Applicability of MMSC:

- Decision version of the problem (whether S'

exists) can be used to check for cover.

- When used as MMSC(U, S, 1)

i.e. limit number of occurrences of u \in U

to a single occurrence, we effectively transform problem to Exact Cover variant (which matches our ideal case exactly).

Another variant worth mentioning is Partial Set Cover, where we again are looking for S'

of minimal cardinality (just as we do in MSC) which covers at least k

elements from universe U

.

Def (Partial Set Cover):

PSC(U, S, k) \to S'

Consider the same set system as in MSC, with additional parameter k \in [1..m]

. Then, Partial Set Cover

(PSC) is the problem of finding S' \subseteq S

of minimal cardinality, that covers at least k

elements of U

.

Partial Set Cover (PSC) vs Maximum Coverage (MC)

: PCS differs from Maximum Coverage problem in a subtle way: in the MC we limit number of subsets |S'| \le k

, for maximum covered elements in U

; in PSC we limit upper bound on how many items are covered |\bigcup\limits_{T \in S'}T| \le k

with S'

of minimal cardinality.

Applicability of PSC:

- Again, decision version can be useful, to check the boundaries (gradually increasing k

) of

S'

existence. With k = |U|

we effectively have MSC problem. In order for PSC be really useful,

we also need to constrain number of occurrences of u \in U

within S'

elements i.e. so that all subsets in S'

are pairwise disjoint.

Exact Cover

The Exact Cover problem is one of Karp's 21 NP-Complete problems.

When exact cover exists within a given set system, the Exact Cover abstracts attestation aggregation perfectly. The problem is that perfectly non-overlapping partitions of U

are not naturally happening in our system (so making them happen can be an attack vector when solving the problem

).

Def (Exact Cover):

EC(U, S) \to S'

Let U

be a finite set of objects, where |U| = n

. Furthermore, let S = {S_1, ..., S_m | S_i \subseteq 2^U}

be a collection of its subsets, where \bigcup_{i = 1}^m S_i = U

. Then, Exact Cover

(EC) is the problem of covering U

with a subset S' \subseteq S

s.t S'_i \cap S'_j = \emptyset, \forall i, j \in [1..m], i \ne j

.

This NP-Hard problem has a nondeterministic backtrack solver algorithm (Algorithm X by D.Knuth). The Algorithm X is capable of finding all

the optimal solutions to the problem.

However, having such an S

that there exists a subcollection of pariwise disjoint subsets that cover U

completely is a rare luck in our system. More than often S

will not contain the solution to EC. In such cases, we still want some partial solution, even if only part of attesters can be collected within a single aggregation.

So, adding constraint similar to MMSC (where we limited number of times u \in U

can occur in S'

), we need to transform the problem into accepting another parameter k \in [1..n]

, with the purpose of finding the S'

, where |\bigcup\limits_{T \in S'}T| \ge k

i.e. union of elements of found subsets covers at least k

elements of U

. Then by gradually increasing k

we want it to be as close to |U|

as possible (max k-cover?

).

Applicability of EC:

- If solution exists, then Algorithm X (effectively implemented using DLX) can find it. If full solution is impossible, we need to explore possibility of finding partial cover.

Maximum Coverage

Def (Maximum Coverage):

MC(U, S, k) \to S'

Let U

be a finite set of objects, where |U| = n

. Furthermore, let S = {S_1, ..., S_m | S_i \subseteq 2^U}

be a collection of its subsets, where \bigcup_{i = 1}^m S_i = U

. Then, Maximum Coverage

(MC) is the problem of finding S' \subseteq S, |S'| \le k

covering U

with maximum cardinality, that's

|S'| \le k \land \mathop{argmax}{*S'*}|*bigcup*{T \in S'}T|

Applicability of MC:

- With additional requirement of S_i \cap S_j, \forall i, j \in [1..m], i \ne j

(pairwise disjoint sets in S'

) we can have a very useful mechanism to build approximate solutions using greedy approach.

Summary and Further Work

So, possible solutions can be enumerated as following:

- Exact Cover (EC)
- Can be used to check for solutions if situations when perfect solution exist are not rare.
- If combined with Partial Set Cover (PSC) for partial cover solutions, can match Attestation Aggregation perfectly.
- Can be used to check for solutions if situations when perfect solution exist are not rare.
- If combined with Partial Set Cover (PSC) for partial cover solutions, can match Attestation Aggregation perfectly.
- Maximum Coverage (MC)

- Greedy algorithm + additional constraint of disjoint sets in S'
- Gradual increase of k

(1 \to |S|

) to obtain maximal cover for a maximum number of available attestations.

- Greedy algorithm + additional constraint of disjoint sets in S'
- Gradual increase of k

(1 \to |S|

) to obtain maximal cover for a maximum number of available attestations.

The scope of this work is to find good enough heuristic to solve the Aggregation Attestation problem as defined in problem statement. There can be a highly effective ways to aggregate attestations that rely on how data is transmitted i.e. instead of concentrating on covering arbitrary set systems, we try to come up with heuristic that will result in a preferable attestations propagating the network (see Heuristically Partitioned Attestation Aggregation for a very interesting approach). Such or similar optimization will eventually be applied, but those are beyond the scope of this effort.