Spectra: Robust Estimation of Distribution Functions in Networks

Miguel Borges, Paulo Jesus, Carlos Baquero, and Paulo Sérgio Almeida

HASLab / INESC TEC & Universidade do Minho
Campus de Gualtar, 4710-057 Braga, Portugal
{mborges,pcoj,cbm,psa}@di.uminho.pt

Abstract. The distributed aggregation of simple aggregates such as minima/maxima, counts, sums and averages have been studied in the past and are important tools for distributed algorithms and network coordination. Nonetheless, this kind of aggregates may not be comprehensive enough to characterize biased data distributions or when in presence of outliers, making the case for richer estimates.

This work presents *Spectra*, a distributed algorithm for the estimation of distribution functions over large scale networks. The estimate is available at all nodes and the technique depicts important properties: robustness when exposed to high levels of message loss, fast convergence speed and fine precision in the estimate. It can also dynamically cope with changes of the sampled local property and with churn, without requiring restarts. The proposed approach is experimentally evaluated and contrasted to a competing state of the art distribution aggregation technique.

1 Introduction

The amount of scientific work is relatively scarce in what concerns expressive aggregation metrics. A recent proposal within this domain (*Adam2*) [11] claims to obtain estimates with a better precision than in previous approaches. It is an algorithm for the estimation of discrete cumulative distribution functions; however, it is not fault tolerant and is also not sensible to the continuous variation of the sampled properties.

Having this as a starting point, we present *Spectra*, a distributed algorithm for the estimation of distribution functions over large scale networks. Its core advantages are resilience to message loss, high convergence speed and high precision of the estimate. It also supports changes of the sampled property and churn. All this is achieved without requiring the protocol to be restarted.

Next, we present a short overview of the state of the art work on the context of distribution aggregation. In Section 3 we briefly state the system model used on this work. Afterwards, we presents the *Spectra* algorithm. We show the evaluation results in Section 5, contrasting them with *Adam2*, the approach that most resembles our own. The final section draws conclusions on the work and presents a few perspectives about future research directions. An extended version of this paper can be found in [2].

K.M. Göschka and S. Haridi (Eds.): DAIS 2012, LNCS 7272, pp. 96–103, 2012.

2 Related Work

Existing aggregation techniques can be divided in different classes, providing different characteristics in terms of performance (time and message load) and robustness, mainly as: hierarchy-based (or tree-based), averaging (or gossip-based), sketches and sampling approaches. A wide and comprehensive overview of the current state of the art on distributed aggregation algorithms is provided in [9].

Some algorithms [4,3] collect samples and apply an estimation method to obtain a rough approximation of the size of a membership. This type of scheme is lightweight in terms of message load, as only a partial number of nodes participate in the sampling process, but is also inaccurate and produces the result at a single node. Another interesting alternative is provided by *averaging* techniques [7,8,1], which can reach an arbitrary accuracy, with the estimate at all nodes converging to the correct result over time.

Most of the existing approaches allow the distributed computation of aggregation functions, and therefore the calculation of scalar values that can result from the combination of those functions. However, they are unable to compute complex aggregates which provide a richer information about some property.

A first gossip-based distribution estimation approach was proposed in [5], randomly exchanging and merging finite lists of bins (i.e. pairs with value and respective counter) between nodes. This algorithm allows data to reach all nodes through multiple paths (and in this sense improving the robustness), but also gives rise to the occurrence of duplicates that will bias the produced estimate.

Adam2 [11] is a recent gossip based approach to approximate distributions, more precisely CDF. Adam2 uses a classic averaging technique, Push-Pull Gossiping [7], and intuitively it can be simply described as the simultaneous execution of multiple instances of this protocol. As will be showed in Section 5, Adam2 inherits the "mass loss" problems of Push-Pull Gossiping, not converging to the correct result even in fault-free scenarios.

3 System Model

Our model assumes the existence of a large number of distributed nodes. Our goal is to estimate an accurate distribution of an attribute present in each node, with a robust aggregation strategy. The assumptions stated below are defined for the purpose of evaluating the system.

The network is modeled as a connected undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, with the set $\mathcal{V}$ representing nodes and the set $\mathcal{E}$ being bidirectional communication links between nodes. We represent the set of adjacent nodes of node i by $\mathcal{D}_i$.

The algorithm is executed synchronously as described in [10](Chapter 2). Each node executes two procedures in lockstep each round: they begin execution by generating messages and sending them to neighbors. Afterwards, nodes compute their new state as a function of their current state, the observed value and received messages from neighbors. We de not require global Ids but only to distinguish the members of the set of neighbors.

Message loss is taken into consideration and modeled as follows: per round, each sent message can be dropped according to a predefined uniform random probability. Dynamic changes in the input values and churn occur at the beginning of each round (i.e., before the message generation step).

4 Spectra – Robust Distribution Estimation

In this section we describe a novel distributed algorithm, *Spectra*, to estimate the distribution of a global attribute, more specifically its Cumulative Distribution Function (CDF). A CDF can be approximated by a mapping from points to the frequencies of the values that are less than or equal to each point. More precisely, considering n nodes and an input value x_i at each node i, the CDF of x can be approximated by a set of k pairs (s_j, e_j), with $0 \leq j < j$, where s_j is an interpolation point and e_j is the fraction of values less or equal than s_j (i.e., $e_j = |\{x_i \mid x_i \leq s_j\}| / n$).

Considering each pair (s_j, e_j) in the CDF, it is possible to estimate e_j through an averaging algorithm: setting 1 as the input value of each node i that satisfies the condition $x_i \leq s_j$ and 0 otherwise, the average of these input values will be the fraction of nodes that fulfil the previous condition, i.e., e_j. This means that each e_j can result from the execution of a distributed averaging algorithm.

The main idea of the proposed algorithm is the combination of this observation with the adaptation of a robust distributed averaging algorithm, Flow Updating [8] (FU)[1], to work with vectors instead of scalars, one component of the vector for each point of the CDF. Simultaneously, whilst the algorithm is converging, a distributed computation of the global minimum and maximum of the values is performed to determine the interval in which the k points of the CDF are estimated.

The computation performed at each node i is detailed by Algorithm 1. The algorithm adapts FU averaging to use vectors instead of scalars. Namely, the flows F_i map for each neighbor a vector of flows (one for each point in the CDF); $\boldsymbol{v_i}$ is a vector which contains the contribution of the node according to the input value x_i, being used as the input to the FU algorithm; and the estimation function yields a vector of estimates, the k points of the CDF.

The algorithm does not assume knowledge of the global minimum and maximum values. Instead, each node keeps a local knowledge of the minimum of maximum known so far in the interpolation interval state variable (I_i). The interval is sent in messages to neighbors, which merge the received intervals. After d rounds, where d is the network diameter, each node i will have the the global minimum and maximum values in the I_i variable.

[1] Flow Updating is a distributed averaging algorithm that computes averages with the value each node observes. It exchanges estimates and flows with its neighbors. Flows are symmetric between two adjacent nodes and quantify the amount a recipient node should adjust its estimate in order to converge to the sender estimate. Eventually, all nodes converge to the same average.

Algorithm 1: Spectra: Algorithm to estimate CDF in distributed networks

1 **inputs:**
2 x_i, value to aggregate
3 $\mathcal{D}_i$, set of neighbors
4 k, number of interpolation points
5 **state variables:**
6 flows: initially, $F_i = \{\}$ `/* mapping from neighbors to flow vectors */`
7 base frequency vector: initially, $\boldsymbol{v_i} = [1 \mid 0 \leq j < k]$
8 interpolation interval: initially, $I_i = (x_i, x_i)$
9 **message-generation function:**
10 $\text{msg}_i(F_i, \boldsymbol{v_i}, I_i, j) = (i, I_i, \boldsymbol{f}, \text{est}(\boldsymbol{v_i}, F_i))$;
11 **with** $\boldsymbol{f} = \begin{cases} F_i(j) & \text{if } (j, \_) \in F_i \\ \boldsymbol{0} & \text{otherwise} \end{cases}$
12 **state-transition function:**
13 $\text{trans}_i(F_i, \boldsymbol{v_i}, I_i, M_i) = (F_i', \boldsymbol{v_i}', I_i')$
14 **with**
15 $I_i' = \text{merge}(I_i \cup \{I \mid (\_, I, \_, \_) \in M_i\})$
16 $\boldsymbol{v_i'} = [\text{if } x_i \leq I_i'(j) \text{ then 1 else 0} \mid 0 \leq j < k]$
17 $F = \{j \mapsto -\text{transform}(\boldsymbol{f}, I, I_i') \mid j \in \mathcal{D}_i \wedge (j, I, \boldsymbol{f}, \_) \in M_i\} \cup$
18 $\quad \{j \mapsto \text{transform}(\boldsymbol{f}, I_i, I_i') \mid j \in \mathcal{D}_i \wedge (j, \_, \_, \_) \notin M_i \wedge (j, \boldsymbol{f}) \in F_i\}$
19 $E = \{i \mapsto \text{est}(\boldsymbol{v_i'}, F)\} \cup$
20 $\quad \{j \mapsto \text{transform}(\boldsymbol{e}, I, I_i') \mid j \in \mathcal{D}_i \wedge (j, I, \_, \boldsymbol{e}) \in M_i\} \cup$
21 $\quad \{j \mapsto \text{transform}(\text{est}(\boldsymbol{v_i}, F_i), I_i, I_i') \mid j \in \mathcal{D}_i \wedge (j, \_, \_, \_) \notin M_i\}$
22 $\boldsymbol{a} = (\sum\{\boldsymbol{e} \mid (\_, \boldsymbol{e}) \in E\})/|E|$
23 $F_i' = \{j \mapsto \boldsymbol{f} + \boldsymbol{a} - E(j) \mid (j, \boldsymbol{f}) \in F\}$
24 **estimation function:**
25 $\text{est}(\boldsymbol{v}, F) = \boldsymbol{v} - \sum\{\boldsymbol{f} \mid (\_, \boldsymbol{f}) \in F\}$
26 **interval merging:**
27 $\text{merge}(S) = (\min(\{l \mid (l, \_) \in S\}), \max(\{u \mid (\_, u) \in S\}))$
28 **interval interpolation:**
29 $(l, u)(j) = l + j \times (u - l)/(k - 1)$
30 **vector transformation function:**
31 $\text{transform}(\boldsymbol{u}, I, I') = [\boldsymbol{u}(\max(\{0\} \cup \{l \mid 0 \leq l < k \wedge I(l) \leq I'(j)\})) \mid 0 \leq j < k]$

Spectra computes an equi-width approximation of the CDF at k equidistant points in the interval between the global minimum and maximum (other variants are possible). We use a notation where an interval $I = (l, u)$ is indexed, i.e., $I(j)$, with j from 0 to $k - 1$, resulting in the k equidistant points of interest from the lower to the upper value (line 29). In this paper we assume that the number of points, k, is fixed and known to all nodes. It is possible to relax this assumption and derive a system where k can be adapted at execution time.

Before global minimum and maximum convergence, vectors calculated at each node or in different rounds refer to different points. At each iteration, as a new interval is computed by merging intervals in messages, *Spectra* needs to transform both the received vectors as well as the vectors from the previous iteration, so that they are meaningful for the new (and potentially different) set of k points. For that, all vectors involved in the execution are transformed from their old to the new interval through the vector transformation function (line 31). This function implements a simple heuristic to obtain the new vector, using the value corresponding to the largest point not greater than the new point (or the first in the vector if no such point exists). Also, the vector of input values to FU, $\boldsymbol{v_i}$, is calculated at each round according to the new interval (line 16).

At each round, in the message generation function (lines 9–11), a single type of message is sent, containing the self id i, its interpolation interval I_i, the flows vector $\boldsymbol{f}$ for each current neighbor j. Sent flows are set according to the current state and otherwise (initially or when a node is added) to 0.

The state-transition function (lines 12–23) takes state (i.e., flows F_i, the node's base frequency vector $\boldsymbol{v_i}$, interpolation interval I_i) and the set of messages M_i received by the node and returns a new state (i.e., flows F_i', base frequency vector $\boldsymbol{v_i'}$ and interpolation interval I_i'). It computes the new interpolation interval setting the lower bound with the minimum of the received minima and does the upper bound with the maximum (line 27). The base frequency vector $\boldsymbol{v_i'}$ is computed from the new interpolation interval I_i' and the initial value x_i. Then, the averaging steps are executed according to FU taking care to transform the involved vectors in order to apply the averaging process to matching interpolation points. These steps result in the creation of the new flows.

The self-adapting nature of the core averaging algorithm, Flow-Updating, on *Spectra* enables it to cope with the dynamic adjustment of all involved vectors. In particular, *Spectra* supports dynamic network changes (i.e., nodes arriving/leaving), simply by adding/removing the flow data associated to neighbors. Moreover, it is also able to seamlessly adapt to changes of the input value x_i – in this case simply by recomputing the vector $\boldsymbol{v_i}$. This is sufficient to allow *Spectra* to operate in settings where the global maximum and minimum do not change.

If the extreme values change due to dynamism, especially if the maximum decreases and the minimum increases, the algorithm as it is will not produce wrong results, but over an overly wide interval. To tighten the interval to the range between the new minimum and maximum, further modifications are due.

At each node i, the estimated CDF at the k equidistant points in the interval I_i is obtained by the estimation function (i.e., $\text{est}(\boldsymbol{v_i}, F_i)$). Over time, the estimated frequency associated to each point converges to the correct value. This is confirmed by the results obtained from evaluation (see Section 5).

5 Evaluation and Comparison with an Existing Approach

The results presented in this work have been obtained using a custom made simulator that implements the model defined in Section 3. We used two error

metrics to quantify the fitness of the estimate to the underlying distribution. The basis of these metrics is the Kolmogorov–Smirnov statistic , as presented in Equation 1. The metric is computed at every round r, for each node n. For every label l, the measure is given by the difference between the cumulative value of the real distribution and the cumulative value of the estimated distribution on node n at round r.

$$KS_r^n = \max_{l \in Labels} \mid P(X \leq l)_r - P(\widehat{X \leq l})_r^n \mid \tag{1}$$

$$KS_{maxr} = \max_{n \in N}(KS_r^n) \tag{2}$$

$$KS_{\mu r} = \frac{1}{\mid N \mid} \sum_{n \in N} KS_r^n \tag{3}$$

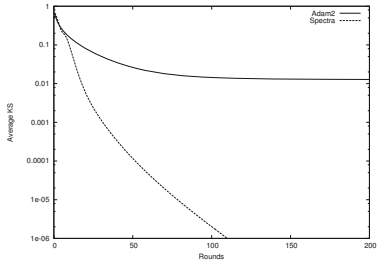
We use two global metrics for the whole network: one to reflect the worst node (Equation 2) and another to reflect the average error (Equation 3). Both equations are computed at every round r.

In order to compare our approach with *Adam2* [11] we assume that both minimum and maximum are previously known to all nodes and the sampling points are evenly distributed between minimum and maximum. The underlying initial values follow a Normal distribution with mean 10 and variance 2. Results were averaged from 30 trials for each scenario.

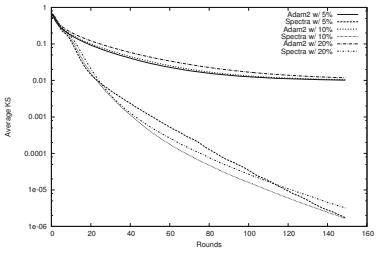
Convergence under no faults. Figure 1(a) presents a graph with the average Kolmogorov-Smirnov distance to the real distribution (as per Equation 3), showing the performance of both algorithms. *Adam2*, whose protocol is based on the *Push-Pull gossiping* averaging algorithm [6], presents a few drawbacks stemming from the fact that it behaves poorly under message loss. One can observe that *Adam2* converges asymptotically to a non-zero value with a continuous offset error while *Spectra* converges indefinitely to zero. Also, the convergence speed is notoriously higher in *Spectra*, with orders of magnitude smaller error.

Fault tolerance. In this scenario we have simulated message loss rates of 5%, 10% and 20% in each round. Results are presented in 1(b). Regarding *Spectra*, we notice a slower overall convergence rate with 5% message loss when compared to the other loss rates. This indicates that the algorithm is not only resilient to message loss but also that with a message loss rate of up to 20%, the convergence rate improves; this contradicts intuition but it is coherent with the behavior of the underlying FU algorithm, observed in [8]. We have also applied *Adam2* to the same rates of message loss, resulting in a systematic offset error.

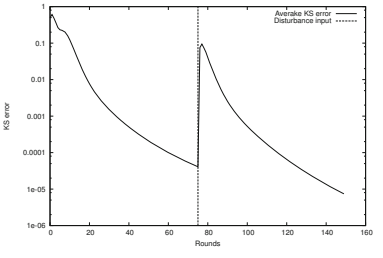
Dynamic adaptation to changes. In order to evaluate the algorithm's behavior under dynamic changes in the sampled value, we have conducted a simulation that introduces a disturbance on 20% of the nodes chosen uniformly random at round 75. The disturbance increased sampled values in 10%, with care not to change the maximum of the network. The obtained results illustrate the adaptive nature of the proposed algorithm (see Figure 1(c)).



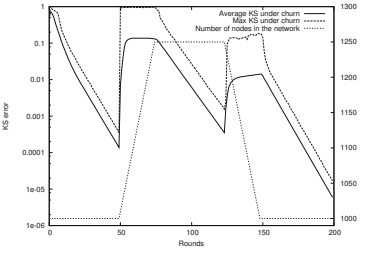
(a) Average KS error rate on a 1000 nodes random network comparing *Spectra* and *Adam2* convergence.



(b) Average KS error rate on a 1000 nodes random network w/ message loss effect on *Spectra* and *Adam2*.



(c) Average KS error on a 1000 nodes random network running *Spectra* with disturbance on initial values.



(d) Average KS error, maximum KS error and node count on a random network running *Spectra* while subjected to churn.

Fig. 1. Comparison w/ *Adam2* (a), (b); and showing dynamic adaptability (c), (d)

Resilience to churn. In order to test the resilience of *Spectra* to churn, We have submitted the algorithm to the departure and arrival of new nodes. It starts with a network of 1000 nodes and at round 50 the number of nodes starts to linearly increase (a 1% increase per round), up to 1250 nodes at round 75. Then, after 50 rounds of stability, nodes randomly leave the network at the same rate until it reaches again 1000 nodes. Node departure is equivalent to nodes crashing, as they leave silently without notifying any neighbors. In order to prevent network partitioning, the average node degree has been increased to 7 (i.e., $\approx ln(1000)$), following what has been done in [8]. Data presented in Figure 1(d) depicts the average KS (as per Equation 3) and maximum KS error (as per Equation 2) for the whole procedure. It also depicts a profile with the number of nodes that constitute the network throughout the rounds.

6 Conclusions and Future Work

We have presented *Spectra*, a distributed algorithm for the estimation of cumulative distribution functions over large scale networks. *Spectra* overcomes the

problems that previous approaches exhibited. It converges to the correct distribution, even when facing high levels of message loss and churn. It also dynamically adapts to changes in the monitored values (and their distribution), and avoids the need to re-start the algorithm and lose progress.

As future work we intend to evolve the technique in order to allow for variations in the minimum and maximum of the target property. We also plan to address strategies for sample point placement, that will be no longer uniform across the min-max range, as this will permit increased sampling in areas where changes in the property are more expressive, and thus further increase precision.

References

1. Almeida, P.S., Baquero, C., Farach-Colton, M., Jesus, P., Mosteiro, M.A.: Fault-Tolerant Aggregation: Flow-Updating Meets Mass-Distribution. In: Fernàndez Anta, A., Lipari, G., Roy, M. (eds.) OPODIS 2011. LNCS, vol. 7109, pp. 513–527. Springer, Heidelberg (2011)
2. Borges, M., Jesus, P., Baquero, C., Almeida, P.S.: Spectra: Robust estimation of distribution functions in networks. Tech. Rep. CoRR abs/1204.1373v1, HASLab / INESC TEC & Universidade do Minho (2012), `http://arxiv.org/abs/1204.1373v1`
3. Cheng, S., Li, J., Ren, Q., Yu, L.: Bernoulli Sampling Based (ε, δ)-Approximate Aggregation in Large-Scale Sensor Networks. In: 29th IEEE Conference on Information Communications (INFOCOM), pp. 1–9 (2010)
4. Ganesh, A.J., Kermarrec, A.M., Le Merrer, E., Massoulié, L.: Peer counting and sampling in overlay networks based on random walks. Distributed Computing 20(4), 267–278 (2007)
5. Haridasan, M., van Renesse, R.: Gossip-based Distribution Estimation in Peer-to-Peer Networks. In: 7th International Conference on Peer-To-Peer Systems (IPTPS). USENIX Association (2008)
6. Jelasity, M., Montresor, A.: Epidemic-style proactive aggregation in large overlay networks. In: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS 2004), pp. 102–109. IEEE Computer Society (2004)
7. Jelasity, M., Montresor, A., Babaoglu, O.: Gossip-based aggregation in large dynamic networks. ACM Transactions on Computer Systems 23(3), 219–252 (2005)
8. Jesus, P., Baquero, C., Almeida, P.S.: Fault-Tolerant Aggregation for Dynamic Networks. In: 29th IEEE Symposium on Reliable Distributed Systems, pp. 37–43 (2010)
9. Jesus, P., Baquero, C., Almeida, P.S.: A Survey of Distributed Data Aggregation Algorithms. Tech. Rep. CoRR abs/1110.0725, HASLab / INESC TEC & Universidade do Minho (2011), `http://arxiv.org/abs/1110.0725`
10. Lynch, N.A.: Distributed algorithms, pp. 17–23. Kaufmann Publishers Inc. (1996)
11. Sacha, J., Napper, J., Stratan, C., Pierre, G.: Adam2: Reliable distribution estimation in decentralised environments. In: 2010 IEEE 30th International Conference on Distributed Computing Systems (ICDCS), pp. 697–707 (June 2010)