Discovering Conservation Rules

Lukasz Golab [#1], Howard Karloff [*], Flip Korn [*], Barna Saha [*], Divesh Srivastava [*]

[#] *University of Waterloo, Canada*
lgolab@uwaterloo.ca

[1] Work done while the author was at AT&T Labs - Research

[*] *AT&T Labs - Research, Florham Park, NJ, USA*
{howard,flip,barna,divesh}@research.att.com

***Abstract*—Many applications process data in which there exists a "conservation law" between related quantities. For example, in traffic monitoring, every incoming event, such as a packet's entering a router or a car's entering an intersection, should ideally have an immediate outgoing counterpart. We propose a new class of constraints—Conservation Rules—that express the semantics and characterize the data quality of such applications. We give confidence metrics that quantify how strongly a conservation rule holds and present approximation algorithms (with error guarantees) for the problem of discovering a concise summary of subsets of the data that satisfy a given conservation rule. Using real data, we demonstrate the utility of conservation rules and we show order-of-magnitude performance improvements of our discovery algorithms over naive approaches.**

I. Introduction

Infrastructure networks are ubiquitous in the modern world: road and rail networks transport people and goods; telecommunication networks are essential for the transmission of multimedia data; and electricity networks deliver power from suppliers to consumers. Given their importance, these networks are continuously monitored over time. For example, highway monitoring systems use road sensors to identify traffic buildup and suggest alternate routes. Similarly, routers in an IP telecommunications network maintain counters to keep track of the traffic flowing through them; these are continuously measured to troubleshoot customer problems, monitor network performance and understand provisioning requirements. Also, the smart grid overlays an electricity network with an advanced metering system, in which power meters measure the electricity flowing through various components of the system; these can be used to dynamically respond to changes in the electricity supply and demand.

In this paper, we focus on the problem of data quality analysis when monitoring infrastructure networks. Missing or delayed data, especially over large time intervals, can be detrimental to any attempt to ensure a reliable, well-functioning network. However, ensuring high quality data is difficult when monitoring large, complex networks. For example, IP network monitoring typically uses the UDP protocol, so measurements can be delayed (or even lost) when there is high network congestion. Another problem occurs if a new router interface is activated and traffic is flowing through it, but this interface is not known to the monitoring system; in this case, there is missing data that is hard to detect. Similar data quality issues arise when monitoring road networks in the presence of sensor failures or unmonitored road segments, when monitoring electricity networks in the presence of hacked power meters or if someone is diverting (stealing) electricity, etc.

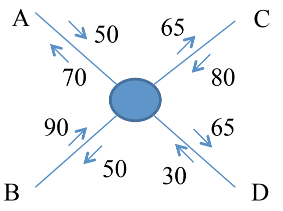


Fig. 1. A network node with counts of incoming and outgoing traffic

A. Conservation Laws

A common approach to data quality analysis is to express the intended semantics using constraints. For example, functional dependencies describe data consistency [7] and inclusion dependencies capture data completeness [2]. Violations of the intended semantics may indicate data quality problems. Inspired by this approach, we seek to identify quality problems in data obtained from network monitoring based on the observations that (i) there often exists a *conservation law* between related quantities in monitored data, and (ii) violations of this law may indicate problems in the monitored data or in the underlying network.

A well-known example of a conservation law in electricity networks is Kirchhoff's node law of conservation of electricity: the current flowing into a node of an electrical circuit equals the current flowing out of the node. Similar laws are expected to hold in road network monitoring (every car that enters an intersection should exit it) and in telecommunication networks (every packet entering a router should exit it).

Figure 1 shows a node (e.g., a road intersection or an IP router) with four bidirectional links (A through D) whose inbound and outbound traffic counts at a point in time are shown. While individual links may carry more inbound than outbound traffic (or vice versa), at the node level we should *ideally* observe the same amount of total in- and out-traffic at all times, as is the case for the node shown in Figure 1. In practice, however, delays can arise in the underlying network, violating this ideal conservation law. For example, in the presence of traffic congestion, a vehicle entering an intersection may exit the intersection with some delay, leading to different amounts of total in and out traffic monitored at any given time. Data quality issues may also cause prolonged violations. For example, in IP network monitoring, if the measurements for link D were lost (e.g., due to the use of UDP), we would see 220 total inbound and 185 total outbound packets; a similar problem occurs if a network engineer turns on link D but does not start monitoring its data. The techniques proposed in this paper can efficiently identify nodes and time intervals where such problems occur in large network monitoring data sets.

B. *Problem Statement and Contributions*

In this paper, we formulate Conservation Rules (CRs) to express and verify conservation laws. Given individual inbound and outbound events (such as a packet's entering or leaving a router, or a vehicle's entering or leaving a road intersection), one could match up event pairs and report, say, average delay and loss, as quality metrics for the conservation rule. However, in IP network monitoring, it is infeasible with respect to storage and processing costs to collect individual packets of a high-speed data stream, making it impossible to find one-to-one correspondences between inbound and outbound events. Therefore, monitoring systems aggregate the total number of inbound and outbound packets at regular intervals. Similarly, road sensors only report aggregated counts.

We therefore assume that we are given two numerical sequences of "inbound" and "outbound" *counts*. Since conservation laws are abstract ideals against which real data may be calibrated, our first contribution is to define *confidence* metrics for CRs, i.e., the extent to which the underlying conservation law holds (Section II). Our metrics estimate the average delay (modeling loss as unbounded delay) between corresponding inbound and outbound events from aggregated counts.

A useful data quality tool for our motivating applications is to discover where in the data a given CR is true or false (up to some degree of confidence). One option is to find the time points with the highest divergence between the inbound and outbound counts. However, this leads to false negatives (it ignores violations that build up slowly) and false positives (large divergences may cancel each other out after a small delay). Another option is to report all sliding windows of some predetermined lengths with the highest divergence between the total inbound and total outbound counts. While more concise than the pointwise approach, it is susceptible to similar problems; for example, a large number of inbound packets at the end of a sliding window whose outbound packets show up in the next time interval can lead to a false positive.

Instead, we adapt the concept of a *tableau* from [2] to summarize how various subsets of the data fit the given CR. A *hold tableau* is a collection of possibly overlapping intervals within the input sequences, each of confidence at least a user-supplied threshold $\hat{c}$. A *fail tableau* is a collection of intervals, each having a confidence at most $\hat{c}$. The TABLEAU DISCOVERY PROBLEM for CRs is to find a (hold or fail) tableau for the given input that contains the fewest possible intervals satisfying confidence, the union of which covers at least a user-supplied fraction of the data.

There are two issues in tableau generation for CRs: identifying candidate intervals satisfying (resp., failing) $\hat{c}$ and constructing minimal tableaux using these intervals. The second issue corresponds to PARTIAL SET COVER for intervals; prior work gives an efficient greedy algorithm that chooses an approximately smallest collection of intervals, each satisfying confidence, whose union is at least some minimum size [12]. The first issue is specific to CRs and requires new algorithms, which we propose in this paper.

Let n be the length of the inbound and outbound sequences given as input. A *candidate interval* $[i, j]$, where $1 \leq i \leq j \leq n$, is one whose confidence, which we define in Section II, is at least (resp., at most) $\hat{c}$. A naive algorithm tests all $\Theta(n^2)$ intervals. As our second contribution, we propose approximation algorithms that run in $O(\frac{1}{\epsilon} n \log n)$ time (under a reasonable assumption to be given in Section III), and return intervals with confidence at least a slightly lower threshold, $\hat{c}/(1+\epsilon)$, for hold tableaux, or at most a slightly higher threshold, $(1+\epsilon)\hat{c}$, for fail tableaux (Section III). Moreover, while the algorithm-produced intervals for hold tableaux, which have confidence at least $\hat{c}/(1+\epsilon)$, may not be maximal among those intervals with confidence at least $\hat{c}/(1+\epsilon)$, they are at least as long as the respective maximal ones with confidence at least $\hat{c}$; an analogous statement holds for fail tableaux.

It turns out that the proposed approximation algorithm's running time is also logarithmically dependent on the area under the curves defined by the cumulative inbound and outbound counts over time. Under reasonable assumptions, this quantity is bounded by a polynomial in n and therefore does not affect the asymptotic running time, but may introduce a larger large constant hidden in the big-O expression. To solve this problem, we present an improved algorithm in Section V whose asymptotic time complexity remains $O(\frac{1}{\epsilon} n \log n)$, but is completely independent of the area under the cumulative count curves.

As our third contribution, we illustrate the utility of CRs and the efficiency of our algorithms on a variety of real data, including network traffic summaries, job submission logs on a computing cluster, credit card data and building entrance/exit data (Section IV). Our experiments show at least an order of magnitude speed-up compared with an algorithm that tests all possible intervals, even with small ϵ, as well as noticeable speed-up of the improved algorithm whose running time does not depend on the area under the cumulative count curves (Section VI). We will also show that our confidence metrics are more suitable for validating conservation laws than those which can be efficiently computed by existing interval-mining algorithms [9].

A preliminary version of this paper, which did not include the improved algorithm from Section V, appeared in the proceedings of the ICDE 2012 conference [**?**].

II. Definitions

Let X and Y be two numeric attributes and let t be an ordered attribute (typically, time). Let $a = \langle a_1, a_2, ..., a_n \rangle$ and $b = \langle b_1, b_2, ..., b_n \rangle$, with $a_i, b_i \geq 0$, be two numeric sequences derived from X and Y, respectively, after ordering with respect to t. We assume uniform spacing in t between adjacent indices i and $i+1$. When a and b are governed by a conservation law, a can be thought of as counting responses to events counted in b (e.g., a = packet departures and b = packet arrivals).

A naive way to measure the adherence of the conservation law within an interval $I = \{i, i+1, ..., j\}$ is to sum up the a_ℓ's and b_ℓ's within I, respectively, and take the ratio. However, this ignores the durations of violations: an outbound event occurring soon after its inbound counterpart would have the same confidence as one which occurs long after its corresponding inbound counterpart. As we shall see, it is natural to infer delay from the cumulative versions of these instantaneous curves.

We define the *derived cumulative time series* A from a where $A_\ell = \sum_{k \leq \ell} a_k$ and $0 = A_0 \leq A_1 \leq \cdots \leq A_n$; B is defined analogously based on b. We assume B dominates A, that is, $B_\ell \geq A_\ell$ for all ℓ. Even if this is not the case, there are various (domain-specific) ways of preprocessing to satisfy this assumption; e.g., setting $A'_\ell := \min\{A_\ell, B_\ell\}$ and $B'_\ell := \max\{A_\ell, B_\ell\}$ for all $1 \leq \ell \leq n$.

Dividing the area under A by the area under B within I accounts for durations of violations. However, the event history before i may have an undesirable impact on the confidence in I. For example, the confidence would be greater than zero even if A stayed flat and B kept increasing in I because the value of A at the start of I may be greater than zero; however, we want the confidence of I to be zero if no outbound events are seen in I, regardless of what happened before I. We propose a confidence measure that discounts this history by taking the ratio of the areas under A and B down to a "baseline," whose height corresponds to the value of A at time $i-1$ (i.e., just before the start of I); see Figure 2(b), which indicates the baseline for $I = [2, 5]$. We now show how this measure naturally arises from estimating the total aggregate delay between inbound and outbound events.

Consider a sequence of seven events, whose inbound (denoted "-in") and outbound (denoted "-out") sightings are plotted in Figure 2(a), with time on the x-axis. Events 1, 2, 5 and 6 incur no delay, while the delays between the incoming and outgoing times of events 3 and 4 are 4 and 3, respectively. Event 7 (whose outbound counterpart has not yet occurred) has a delay of at least one. The total delay is at least eight.

We define the cost of an edge in this pairing (i.e., the delay), from an inbound event at time k to an outbound event at time $k' \geq k$, to be $k' - k$. For the present problem, however, the events are anonymous: there are b_ℓ inbound events and a_ℓ outbound events at each time ℓ, but there is no way to know which inbound event is paired with which outbound event. Morever, events may be dropped or not recorded, so there may be inbound events paired with no outbound event, and vice versa. Nonetheless, we would like to define a confidence measure that pairs up the inbound and outbound events "as much as possible." Assume that there are the same number of inbound and outbound events.

Definition 1 *A* rightward perfect matching *is one that pairs the inbound and outbound events, with each inbound event paired with an outbound event occurring no earlier.*

For example, suppose that the 7-in event did not exist in Figure 2(a); then, pairing up each remaining x-in with the corresponding x-out gives a rightward perfect matching, as does pairing up 1-in with 1-out, 2-in with 2-out, 3-in with 5-out, 4-in with 6-out, 5-in with 3-out and 6-in with 4-out. In the anonymous case, however, there is the question of whether a rightward perfect matching exists, and if it does, what its delay is. In order for a rightward perfect matching to exist, we must have $A_n = B_n$ and $A_\ell \leq B_\ell$ for all ℓ. It is trivial to show (by induction) that this condition is sufficient as well.

Lemma 1 *There exists a rightward perfect matching between inbound and outbound events iff $A_n = B_n$ and $\forall \ell, A_\ell \leq B_\ell$.*

In fact, the total delay of *any* rightward perfect matching is the same (e.g., it equals seven in Figure 2(a) assuming that the 7-in event does not exist).

Lemma 2 *If $A_n = B_n$ and $A_\ell \leq B_\ell$ for all ℓ, then* any *rightward perfect matching between inbound and outbound events has delay $\sum_{\ell=1}^{n}(B_\ell - A_\ell)$.*

Proof sketch. A rightward perfect matching exists by Lemma 1. An edge extending from an event arriving at time k to one occurring at time $k' \geq k$ has delay $k' - k$, so the delay of the perfect matching is $\sum_{k'} a_{k'} \cdot k' - \sum_k b_k \cdot k$. Simple algebra, using the fact that $\sum_{\ell=1}^{n} a_\ell = \sum_{\ell=1}^{n} b_\ell$, shows that $\sum_{\ell=1}^{n} a_\ell \cdot \ell - \sum_{\ell=1}^{n} b_\ell \cdot \ell = \sum_{\ell=1}^{n}(B_\ell - A_\ell)$. ∎

While we are given that $A_\ell \leq B_\ell$ for all ℓ, there is no *a priori* reason to believe that $A_n = B_n$. Solely for the purpose of motivating the definition of confidence, let us (temporarily) increase a_n (and, therefore, A_n) by $B_n - A_n$.

We now consider total delay in an interval $[i, j]$. There is no reason to believe that the number of inbound events in total over I should equal the number of outbound events in the same interval. Assume that all events that occur at or before time i are postponed so as to occur exactly at time i, and that all events that occur at or after time $j+1$ are advanced so as to occur exactly at time $j+1$; all of $A_i, B_i, A_{i+1}, B_{i+1}, ..., A_j, B_j$ remain unchanged. In the rightward perfect matching between inbound and outbound events (which still exists), the delay of an edge from an event arriving at time k now cannot exceed $(j+1) - k$; the delay of an edge to an event occuring at time k' now cannot exceed $k' - i$. Moreover, Lemma 2 proves that the delay of *every* such rightward perfect matching is still $\sum_{\ell=i}^{j}(B_\ell - A_\ell)$. (We should really be including $B_{j+1} - A_{j+1}$, but that term is 0.)

Since we want the confidence to be zero when delay is maximized (i.e., $a_i = a_{i+1} = \cdots = a_j = 0$) and one when

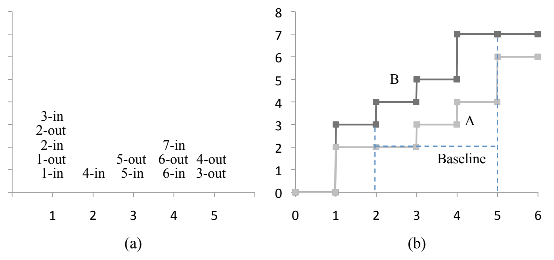


Fig. 2. Sequences of events and their cumulative counts

delay is minimized (i.e., $A_i = B_i, ..., A_j = B_j$), we divide $\sum_{\ell=i}^{j}(B_\ell - A_\ell)$ by $\sum_{\ell=i}^{j}(B_\ell - A_{i-1})$ and subtract this quantity from one, getting $\sum_{\ell=i}^{j}(B_\ell - A_{i-1}) - \sum_{\ell=i}^{j}(B_\ell - A_\ell) = \sum_{\ell=i}^{j}(A_\ell - A_{i-1})$ in the numerator.

DEFINITION 2 *Given cumulative sequences A and B, the* confidence $conf(I)$ *of interval $I = [i, j]$ $(1 \le i \le j \le n)$ is defined (if the denominator is positive) as*

$$\frac{\sum_{l=i}^{j}(A_l - A_{i-1})}{\sum_{l=i}^{j}(B_l - A_{i-1})}.$$

In Figure 2 (b), we illustrate the A and B curves obtained from Figure 2 (a). Here, a = out and b = in, so $a = \langle 2, 0, 1, 1, 2\rangle$, $b = \langle 3, 1, 1, 2, 0\rangle$, $A = \langle 0, 2, 2, 3, 4, 6\rangle$ and $B = \langle 0, 3, 4, 5, 7, 7\rangle$. The confidence of $I = [2, 5)$ is $\frac{3}{10}$, which is the area under A in I down to the baseline (of height A_{i-1}; here, $i = 2$), divided by the area under B in I down to the baseline.

We call this measure the *balance* model (denoted by $conf^b$) because it penalizes for the balance $B_{i-1} - A_{i-1}$ existing just before I begins. In other words, for the purpose of computing confidence, we count the delays of all the unmatched incoming events from the past as if those events had occurred just before I starts. If we want to focus strictly on what happens within I, there are two natural ways of ignoring past events: move the B curve down by an amount equal to $B_{i-1} - A_{i-1}$ so that it coincides with A at the beginning of I, or move the A curve up by the same amount. (Technically, instead of $B_{i-1} - A_{i-1}$, we use $S_i = \min_{i \le k \le n}\{B_k - A_k\}$, which is the smallest difference between A and B in the suffix from i to n, in case some of the outbound events missing at i are delayed and eventually occur. This guarantees that B still dominates A.) We call the former the *debit model* since we are removing unmatched inbound events from B, and the latter the *credit model* since we are injecting missing outbound events into A.

DEFINITION 3 *Given cumulative sequences A and B, the confidence $conf^c(I)$ of interval $I = [i, j]$, with credit S_i applied at i, is defined (if the denominator is positive) as*

$$\frac{\sum_{\ell=i}^{j}(A_\ell - A_{i-1} + S_i)}{\sum_{\ell=i}^{j}(B_\ell - A_{i-1})}.$$

DEFINITION 4 *Given cumulative sequences A and B, the* confidence $conf^d(I)$ *of interval $I = [i, j]$, with debit S_i applied at i, is defined (if the denominator is positive) as*

$$\frac{\sum_{\ell=i}^{j}(A_\ell - A_{i-1})}{\sum_{\ell=i}^{j}(B_\ell - A_{i-1} - S_i)}.$$

In the example from Figure 2, we can compute $conf^d$ of I by shifting B down by one (which is the discrepancy between A and B at time 2). This gives $\frac{3}{7}$. By shifting A up by one, we get $conf^c$ of I, which is $\frac{6}{10}$.

Choosing a confidence model depends on the hypothesis that the analyst wishes to test. The credit model is appropriate if one suspects missing outbound events, the debit model is more appropriate when inbound events may have been spuriously counted, and the balance model is the model of choice if there are problems with both outbound and inbound events. We will give real-life scenarios corresponding to each model in Section IV. For instance, we will analyze data obtained from sensors that count how many people enter and exit a building through the front door. We suspect that there exists an unmonitored side exit, so we use the credit model to account for missing outbound events.

III. ALGORITHMS

We are given two non-negative real sequences, $a = \langle a_1, a_2, \ldots, a_n\rangle$ and $b = \langle b_1, b_2, \ldots, b_n\rangle$. The cumulative time series A and B can be obtained in linear time. It naively takes linear time to compute the confidence of one interval; however, again by simple linear-time preprocessing, confidence can be computed in constant time. Given user-supplied parameters $\hat{c}$ and $\hat{s}$, the TABLEAU DISCOVERY PROBLEM seeks a collection of the fewest possible intervals $[i, j]$ (with $i \le j$ integers) such that (1) each interval satisfies confidence $\hat{c}$ and (2) the union of the intervals in the tableau contains at least $\hat{s}n$ time ticks, where n is the length of the input time series a and b. Here is the general scheme:

1) (Candidate interval generation phase:) For each i, find the largest $j \ge i$ satisfying confidence $\hat{c}$ for $[i, j]$, as defined in either the balance, credit, or debit model, as approriate, provided one such interval exists.
2) (Tableau generation phase:) Apply the greedy PARTIAL SET COVER algorithm from [12] to the intervals selected in step 1 to find a subcollection whose union has size at least $\hat{s}n$. At each iteration, this algorithm chooses an interval that covers the most points that have not already been covered by previously chosen intervals.

The prior work shows that after the intervals are generated in step 1, the greedy PARTIAL SET COVER algorithm is guaranteed to generate a tableau of size at most a small constant factor larger than the optimal tableau, and often only slightly larger. None of our algorithms change step 2 in the least—all the work lies in modifying step 1 suitably. Therefore, we consider the following problem here.

DEFINITION 5 *The* CANDIDATE INTERVAL GENERATION PROBLEM *is to find, for each i, the largest $j \ge i$ (if one exists) such that the interval $[i, j]$ satisfies confidence.*

We examine the running time of an exhaustive algorithm for finding, for each i, the largest j such that $[i, j]$ satisfies confidence. The algorithm considers, for each i, all linearly many intervals $[i, j]$, and computes the confidence for each chosen interval individually to check if it satisfies the threshold $\hat{c}$. The time complexity of this naive exhaustive search is $\Theta(n^2)$: if one examines all $\Theta(n^2)$ different intervals $[i, j]$, over all $i \leq j$, clearly the running time will be $\Omega(n^2)$. For large datasets, running a quadratic-time algorithm can be extremely time-consuming, perhaps infeasible. Therefore, we trade off accuracy for speed. Here is the general framework for generating intervals for hold tableaux:

For each i, do:

1) Carefully generate a sparse set of intervals $[i, j]$.
2) Compute the confidence of each such interval.
3) Output the longest of the selected intervals having confidence at least $\hat{c}/(1 + \epsilon)$, if any exist.

Replacing "at least $\hat{c}/(1 + \epsilon)$" by "at most $\hat{c}(1 + \epsilon)$" gives candidate intervals for fail tableaux.

It is not obvious, in light of the use of a sparse set of intervals with left endpoint i (rather than all), and the use of confidence thresholds $\hat{c}/(1+\epsilon)$ for hold tableaux and $\hat{c}(1+\epsilon)$ for fail tableaux, that the tableaux generated by the algorithm have any relation to the optimal tableaux. We might choose j's to be "geometrically equally spaced" so that if the longest interval of confidence at least $\hat{c}$ with left endpoint i is, say, $[i, j^*]$, then one of the sparse set of intervals tested by the algorithm is an interval $[i, j]$ with j only slightly larger than j^*, so that $[i, j]$ is only slightly longer than $[i, j^*]$. However, since the curves can grow an unbounded amount between time steps, there is no guarantee that the confidence of $[i, j]$ will be at least $\hat{c}/(1+\epsilon)$. The trick will be to select interval endpoints based on the growth of areas under A and B.

We define two additional real sequences, $\langle H^A_1, H^A_2, H^A_3, \ldots, H^A_n\rangle$ and $\langle H^B_1, H^B_2, H^B_3, \ldots, H^B_n\rangle$, in a problem-dependent way:

1) In the balance model, $H^A_i = H^B_i = A_{i-1}$;
2) in the credit model, $H^A_i = A_{i-1} - \min_{k\geq i}\{B_k - A_k\}$ and $H^B_i = A_{i-1}$; and
3) in the debit model, $H^A_i = A_{i-1}$ and $H^B_i = A_{i-1} + \min_{k\geq i}\{B_k - A_k\}$.

These H^A_i and H^B_i values correspond to the baseline values discussed earlier, and hence $A_l - H^A_i \geq 0$ and $B_l - H^B_i \geq 0$ for all $l \geq i$. In all three cases, all the n H^A_i and n H^B_i values can be computed in $O(n)$ time.

Define $area_A(i, j) = \sum_{l=i}^{j}(A_l - H^A_i)$ and $area_B(i, j) = \sum_{l=i}^{j}(B_l - H^B_i)$. Note that the subscript on the H terms is i, not l. Then the confidence $conf([i, j]) = area_A(i, j)/area_B(i, j)$, defined only if the denominator is positive.

Our goal is to make sure we test at least one interval $[i, j]$ with $j \geq j^*$ such that the confidence of interval $[i, j]$ is almost as large (in the case of hold tableaux) as that of $[i, j^*]$. Since the confidence of $[i, j]$ is $area_A(i, j)/area_B(i, j)$, and both areas are, for fixed i, nondecreasing in j (hence $area_A(i, j) \geq area_A(i, j^*)$), it will suffice to have $area_B(i, j) \leq (1 + \epsilon)area_B(i, j^*)$. In that case $conf(i, j) = area_A(i, j)/area_B(i, j) \geq area_A(i, j^*)/area_B(i, j) \geq area_A(i, j^*)/[(1 + \epsilon)area_B(i, j^*)] \geq conf(i, j^*)/(1 + \epsilon) \geq \hat{c}/(1 + \epsilon)$. Since the goal is to have $area_B(i, j) \leq (1 + \epsilon)area_B(i, j^*)$, it will suffice to only compute the confidence of intervals $[i, j]$ with $area_B(i, j)$ growing geometrically by a factor of $1 + \epsilon$. This will be the sparse set of intervals whose confidence we will compute; the number will be logarithmic (to the base $1 + \epsilon$) in the $area_B$'s.

A. *Hold Tableau Interval Selection*

We give *one* simple, generic algorithm for hold tableaux that works for all three models: balance, credit, and debit.

We assume that neither the a sequence nor the b sequence is identically zero. Define Δ to be the minimum positive a_i or b_i. The algorithm is as follows.

For $i = 1, 2, 3, ..., n$ such that $area_B(i, n) > 0$, do:

1) For each integer $l = 0, 1, 2, ..., \lceil\log_{1+\epsilon}(area_B(i, n)/\Delta)\rceil$, find the largest integer $j \geq i$, if one exists, such that $area_B(i, j) \leq \Delta(1 + \epsilon)^l$. Call that largest j, r_{il}.
2) Compute the confidence of each interval $[i, r_{il}]$ for the same l's.
3) Output the longest interval $[i, r_{il}]$ of confidence at least $\hat{c}/(1 + \epsilon)$, if at least one exists.

Like the algorithms to appear later for constructing fail tableaux, the algorithm is scale invariant in that if both the a and b sequences are multiplied by the same positive scalar, neither the answers nor the running time will change.

We now give an example to illustrate the various steps of the algorithm for hold tableau interval selection, which will also aid in the understanding of the proof of Theorem 1. Let $a = \langle 5, 8, 6, 8, 7, 4, 3, 20, 11, 7\rangle$ and $b = \langle 10, 8, 11, 13, 6, 6, 5, 9, 12, 6\rangle$. Thus, $n = 10$ and $\Delta = \min\{\min_{i=1}^{10} a_i, \min_{i=1}^{10} b_i\} = 3$. The two cumulative series are $A = \langle 0, 5, 13, 19, 27, 34, 38, 41, 61, 72, 79\rangle$ and $B = \langle 0, 10, 18, 29, 42, 48, 54, 59, 68, 80, 86\rangle$. So $area_B(1, 10) = 10 + 18 + 29 + 42 + 48 + 54 + 59 + 68 + 80 + 86 = 494$.

Suppose $\epsilon = 1$ (in general ϵ will be much smaller, say, 0.01 or 0.1) and hence, $1 + \epsilon = 2$. We now compute the indices $r_{i1}, r_{i2}, \ldots, r_{i\ell}$ for each $i = 1, 2, \ldots, 10$, where $\ell = \lceil\log_2(area_B(1, n)/\Delta))\rceil = \lceil\log_2(494/3)\rceil = 8$. From the definition of r_{il}s, when $l = h$, we need the largest index $j \geq i$ such that $area_B[i, j] \leq \Delta(1 + \epsilon)^h = 3 \times 2^h$. Thus, for $l = 0$, this threshold is 3, for $l = 1$ the threshold is $3 \times 2 = 6$, and so on. Note that $r_{il} = 0$ implies that the particular value of r_{il} does not exist; apart from these zero values, for each i, r_{il}, $l = 1, 2, \ldots, 8$, is a non-decreasing sequence. This property helps us to achieve the desired running time bound along with the fact that, for any given interval $[i, j]$, $area_A[i, j]$ and $area_B[i, j]$ can be computed in constant time.

Let us illustrate for fixed $i = 3$. First, $area_B[3, 3] = 29 - 13 = 16$ is greater than 3×2^0, so $r_{3,0} = 0$. We have $3 \times 2 = 6$ and $area_B[3, 3] = 16$, so again $r_{3,1} = 0$. Also $3 \times 2^2 = 3 \times 4 = 12 < 16 = area_B[3, 3]$, so $r_{3,2} = 0$ as well. Now $3 \times 8 = 24$

and $area_B[3,3] = 16$ but $area_B[3,4] = 45$, so $r_{3,3} = 3$; $3 \times 16 = 48$ and $area_B[3,4] = 45$ but $area_B[3,5] = 80$ so $r_{3,4} = 4$; $area_B[3,5] = 80 \leq 3 \times 32 = 96 < area_B[3,6] = 121$ so $r_{3,5} = 5$; $area_B[3,7] = 167 \leq 3 \times 64 = 192 < area_B[3,8] = 222$ so $r_{3,6} = 7$; $area_B[3,9] = 289 \leq 3 \times 128 = 384 < area_B[3,10] = 362$ so $r_{3,7} = 9$; and finally $area_B[3,10] = 362 \leq 3 \times 256 = 768$, so $r_{3,8} = 10$. The only intervals $[i, r_{il}]$ considered are $[3,3], [3,4], [3,5], [3,7], [3,9]$, and $[3,10]$. Now $conf(i, r_{i,3}) = conf(3,3) = 6/16 = 0.375$, $conf(i, r_{i,4}) = conf(3,4) = 20/45 = 0.4444$, $conf(i, r_{i,5}) = conf(3,5) = 41/80 = 0.5125$, $conf(i, r_{i,6}) = conf(3,7) = 94/121 = 0.7768$, $conf(i, r_{i,7}) = conf(3,9) = 201/222 = 0.905$, and $conf(i, r_{i,8}) = conf(3,10) = 267/322 = 0.737$. Suppose we set $\hat{c} = 1$, and thus $\hat{c}/(1+\epsilon) = 0.5$. Then the intervals $[3,5], [3,7], [3,9]$ and $[3,10]$ are valid choices. So we select the longest one, $[3,10]$.

THEOREM 1 *The total running time of the given algorithm for the balance, credit, or debit models is $O(n \log_{1+\epsilon}(area_B(1,n)/\Delta))$, which is $O(\frac{1}{\epsilon} n \log_2(area_B(1,n)/\Delta))$ if $\epsilon \leq 1$, and hence $O(\frac{1}{\epsilon} n \log n)$ if $area_B(1,n)/\Delta$ is bounded by a polynomial in n.*

Proof: First, computing $area_A(i,j)$, $area_B(i,j)$, and $conf(i,j)$ are constant-time operations, since we can precompute $S_j = \sum_{l=1}^{j} A_l$ and $T_j = \sum_{l=1}^{j} B_l$. Then
$area_A(i,j) = (S_j - S_{i-1}) - (j-i+1)H_i^A$,
$area_B(i,j) = (T_j - T_{i-1}) - (j-i+1)H_i^B$, and
$conf(i,j) = area_A(i,j)/area_B(i,j)$,
provided the denominator is positive. Now, the only remaining issue is how to compute the r_{il}'s quickly. For this, we make an assumption (which we will later prove) regarding the r_{il}s.

LEMMA 3 *Suppose $r_{i-1,l} \leq r_{il}$ for all i, l when both $r_{i-1,l}$ and r_{il} are defined. Then the total time to compute all r_{il}'s is $O(n \log_{1+\epsilon}(area_B(1,n)/\Delta))$.*

PROOF SKETCH. For brevity, we assume that all r_{il}'s exist and leave the general case as an exercise for the reader.

A candidate integer $t \geq i$ equals r_{il} (when $area_B(i,n) > 0$) if and only if $area_B(i,t) \leq \Delta(1+\epsilon)^l$ and $area_B(i,t+1) > \Delta(1+\epsilon)^l$ (or $t = n$). (We are using implicitly the fact that the smallest nonzero $area_B(i,t)$ is at least Δ.)

We show how, given $l \leq \lceil \log_{1+\epsilon}(area_B(1,n)/\Delta) \rceil$, to compute all n values $r_{1l}, r_{2l}, r_{3l}, ..., r_{nl}$ in a total of $O(n)$ time. (Hence the total running time, over all values of l, will be $O(n \log_{1+\epsilon}(area_B(1,n)/\Delta))$.)

We start with $t = 1$ and increment t until $r_{1l} = t$ is found. When it is, we try to find r_{2l}, starting with $t = r_{1l}$ (which is safe because $r_{1l} \leq r_{2l}$), in each step testing t and incrementing t, until r_{2l} is found. Once it is, we try to find r_{3l}, starting with $t = r_{2l}$ (which is safe because $r_{2l} \leq r_{3l}$), in each step testing t and incrementing t, until r_{3l} is found. We continue in this way, starting the search for $r_{i+1,l}$ at the value r_{il} just found. We continue in this way until r_{nl} is found.

The key point is that t is never decreased. The number of times t is increased cannot exceed n, and in iterations in which t is not increased, we change from seeking r_{il} to seeking $r_{i+1,l}$; this can happen at most n times, making for a total of at most $2n$ iterations. ■

LEMMA 4 *If $H_i^B \geq H_{i-1}^B$ for all i, then $r_{i-1,l} \leq r_{il}$ if both are defined.*

Proof: In general, r_{il} is the maximum j such that $\sum_{l=i}^{j}(B_l - H_i^B) \leq \Delta(1+\epsilon)^l$ and $r_{i-1,l}$ is the maximum j such that $\sum_{l=i-1}^{j}(B_l - H_{i-1}^B) \leq \Delta(1+\epsilon)^l$. Hence $\sum_{l=i}^{r_{i-1,l}}(B_l - H_{i-1}^B) \leq \Delta(1+\epsilon)^l$. If $H_i^B \geq H_{i-1}^B$, then $\sum_{l=i}^{r_{i-1,l}}(B_l - H_i^B) \leq \Delta(1+\epsilon)^l$, from which it follows that $r_{il} \geq r_{i-1,l}$. ■

LEMMA 5 *For the balance, credit, and debit models, $H_i^B \geq H_{i-1}^B$.*

Proof: For the balance and credit models, since $H_i^B = A_{i-1}$, we need simply show that $A_{i-1} \geq A_{i-2}$, which is obvious since A is nondecreasing. For the debit model, $H_i^B = A_{i-1} + \min_{k \geq i}\{B_k - A_k\}$. From the fact that $A_{i-2} \leq A_{i-1}$ and $\min_{k \geq i-1}\{B_k - A_k\} \leq \min_{k \geq i}\{B_k - A_k\}$, we infer that $H_{i-1}^B \leq H_i^B$. ■

Theorem 1 now follows immediately from Lemmas 3, 4, and 5. ■

In principle, the logarithmic dependence on $area_B(1,n)/\Delta$ could be a problem, but this would require $area_B(1,n)/\Delta$ to be huge. If these values were exponential in n, then the logarithm would be linear in n (giving $O(n^2/\epsilon)$ running time), but in this case it would take linear memory even to store *one* of the numbers exactly, and then doing arithmetic on such long numbers would be prohibitively slow. (Representing the numbers as fixed-precision reals obviates this objection, but introduces round-off error into the calculations; we don't know how such round-off errors would affect the final answer.) In most practical cases, once one divides out by the smallest positive a_i or b_i (which is Δ), one would probably expect polynomial growth. In other words, one would probably expect $\log_2(area_B(1,n)/\Delta)$ to be $O(\log_2 n)$, giving overall running time of $O((n \log n)/\epsilon)$. Indeed, in all the data sets we considered, the $\log_2(area_B(1,n)/\Delta)$ factor was very small.

The next theorem states that the intervals produced by the algorithm are accurate.

THEOREM 2
1) *(No false positives) If the algorithm outputs an interval $[i,j]$, then $conf(i,j) \geq \hat{c}/(1+\epsilon)$.*
2) *(No false negatives) Given i, let $j^* \geq i$ be largest such that $conf(i,j^*) \geq \hat{c}$ (if such a $j^* \geq i$ exists). Then the algorithm produces an interval $[i,j']$, $j' \geq j^*$, having confidence at least $\hat{c}/(1+\epsilon)$.*

Proof: The first part is trivial, as it is obvious from the last step of the algorithm that if the output includes an interval $[i,j]$, then $conf(i,j) \geq \hat{c}/(1+\epsilon)$. Modulo the distinction between $\hat{c}$ and $\hat{c}/(1+\epsilon)$, there are no "false positives."

For the second part, define h such that $\Delta(1+\epsilon)^{h-1} < area_B(i,j^*) \leq \Delta(1+\epsilon)^h$. Since r_{ih} is the largest index

j such that $area_B(i,j) \leq \Delta(1+\epsilon)^h$, and $area_B(i,j^*) \leq \Delta(1+\epsilon)^h$, it follows that $r_{ih} \geq j^*$. The algorithm did compute the confidence of interval $[i, r_{ih}]$. We now prove that $conf(i, r_{ih}) \geq \hat{c}/(1+\epsilon)$, and hence the algorithm will report interval $[i, r_{ih}]$ (or a longer interval, also of confidence at least $\hat{c}/(1+\epsilon)$).

$$conf(i, r_{ih}) = \frac{area_A(i, r_{ih})}{area_B(i, r_{ih})} \geq \frac{area_A(i, j^*)}{area_B(i, r_{ih})}.$$

Now $area_B(i, r_{ih}) \leq \Delta(1+\epsilon)^h$ and $area_B(i,j^*) > \Delta(1+\epsilon)^{h-1}$. Therefore $area_B(i, r_{ih})/area_B(i,j^*) < 1+\epsilon$. It follows that

$$conf(i, r_{ih}) \geq \frac{area_A(i,j^*)}{(1+\epsilon)area_B(i,j^*)} = \frac{1}{1+\epsilon} conf(i,j^*)$$

$$\geq \frac{1}{1+\epsilon}\hat{c}. \blacksquare$$

To explain why we want $j' \geq j^*$, suppose an optimal tableau has m intervals, each of the form $[i, j^*]$, each of confidence at least $\hat{c}$, whose union covers at least a specified fraction $\hat{s}$ of $\{1, 2, ..., n\}$. We want to show that the algorithm generates a small tableau, each of whose intervals has (at least) *almost* the same confidence. We may assume that each j^* is largest such that $[i, j^*]$ has confidence at least $\hat{c}$. By the "no-false-negatives" property, the algorithm generates an interval $[i, j']$, $j' \geq j^*$, so that $[i, j']$ *contains* $[i, j^*]$, and hence there exists a tableau *of at most* m intervals $[i, j']$ each produced by the algorithm and each of confidence at least $\hat{c}/(1+\epsilon)$, whose union covers at least $\hat{s}n$ points.

One might ask if the output tableau could be *smaller* than the optimal tableau, as the algorithm is allowed intervals of confidence at least $\hat{c}/(1+\epsilon)$, which is less than $\hat{c}$. The answer is yes. In light of the fact that more concise tableaux are better, and that the confidence of each interval is almost $\hat{c}$, "shorter-than-optimal" tableaux seem desirable.

One can imagine situations, though, in which getting confidence at least $\hat{c}$ (and not just $\hat{c}/(1+\epsilon)$) is crucial. In this case, one can run the algorithm with $\hat{c}' = \hat{c}(1+\epsilon)$ (provided this last expression is at most 1), which will guarantee confidence at least $(\hat{c}(1+\epsilon))/(1+\epsilon) = \hat{c}$ for each output interval. The drawback, of course, is that the size of the optimal tableau built with these intervals can no longer be compared to that of the optimal tableau whose intervals have confidence at least $\hat{c}$; it is instead no greater than that of the optimal tableau whose intervals have confidence at least $(1+\epsilon)\hat{c}$.

B. *Fail Tableau Interval Selection*

For fail tableaux, we want, ideally, to generate intervals $[i, j']$, where j' is largest such that $conf(i, j') \leq \hat{c}$ (as opposed to $conf(i, j') \geq \hat{c}$, in the case of hold tableaux). We instead generate intervals with confidence at most $\hat{c}(1+\epsilon)$. While we now want confidence bounded *above* by $\hat{c}$ (or $\hat{c}(1+\epsilon)$), if the optimal interval covering at least $\hat{s}n$ points is $[i, j^*]$, we still need $j' \geq j^*$, not $j' \leq j^*$. The reason is that, once again, we want to know that if the optimal tableau consists of m intervals, then there is a collection of m algorithm-generated intervals that cover the same number of elements, and that is why the algorithm must generate intervals which contain the intervals in the optimal tableau (and hence $j' \geq j^*$).

Instead of using the generic algorithm above, which involves $area_B(i,j)$, the generic algorithm for fail tableaux uses $area_A(i,j)$. We will need to treat the balance and debit models differently from the credit model. We start with the former two.

C. *Fail Tableaux in the Balance And Debit Models*

Here is the algorithm to choose intervals for fail tableaux in the balance or debit model. The algorithm finds, for each i such that there is an interval $[i, j^*]$ of confidence at most $\hat{c}$, an interval $[i, j']$, $j' \geq j^*$, of confidence at most $(1+\epsilon)\hat{c}$. Unfortunately the case in which $conf(i, j^*) = 0$ requires easy special care. To simplify the exposition, the algorithm below and its corresponding proof assume that $conf(i, j^*) > 0$. As before, let Δ denote the smallest nonzero a_i or b_i.

For $i = 1, 2, 3, ..., n$ such that $area_A(i, n) > 0$, do:

1) For each integer $l = 0, 1, 2, ..., \lceil \log_{1+\epsilon}(area_A(i,n)/\Delta) \rceil$, find the largest $j \geq i$, if one exists, such that $0 < area_A(i,j) \leq \Delta(1+\epsilon)^l$. Call that largest j s_{il}.
2) Compute the confidence of each interval $[i, s_{il}]$ for the same l's.
3) Output the longest interval $[i, s_{il}]$ of confidence at most $(1+\epsilon)\hat{c}$, if at least one exists.

THEOREM 3 1) *(No false positives) If the algorithm outputs an interval* $[i,j]$*, then* $conf(i,j) \leq \hat{c}(1+\epsilon)$*.*
2) *(No false negatives) Given i, let* $j^* \geq i$ *be largest such that* $conf(i, j^*) \leq \hat{c}$ *(if such a* $j^* \geq i$ *exists). Then the algorithm produces an interval* $[i, j']$*,* $j' \geq j^*$*, having confidence at most* $\hat{c}(1+\epsilon)$*.*

Proof: The first part is trivial, as it is obvious from the last step of the algorithm that if the output includes an interval $[i,j]$, then $conf(i,j) \leq \hat{c}(1+\epsilon)$.

For the second part, we assume, *with* loss of generality, that $area_A(i, j^*) > 0$. Define $h \geq 0$ such that $\Delta(1+\epsilon)^{h-1} < area_A(i,j^*) \leq \Delta(1+\epsilon)^h$ (which exists by our assumption that $area_A(i,j^*)$ is positive and hence at least Δ). Since s_{ih} is the largest index j such that $area_A(i,j) \leq \Delta(1+\epsilon)^h$, and $area_A(i,j^*) \leq \Delta(1+\epsilon)^h$, it follows that $s_{ih} \geq j^*$. The algorithm did compute the confidence of interval $[i, s_{ih}]$. We now prove that $conf(i, s_{ih}) \leq \hat{c}(1+\epsilon)$, and hence the algorithm will report interval $[i, s_{ih}]$ (or a longer interval, also of confidence at most $\hat{c}(1+\epsilon)$).

$$conf(i, s_{ih}) = \frac{area_A(i, s_{ih})}{area_B(i, s_{ih})} \leq \frac{area_A(i, s_{ih})}{area_B(i, j^*)}.$$

Now $area_A(i, s_{ih}) \leq \Delta(1+\epsilon)^h$ and $area_A(i,j^*) > \Delta(1+\epsilon)^{h-1}$. Therefore $area_A(i, s_{ih})/area_A(i,j^*) < 1+\epsilon$.

It follows that

$$conf(i, s_{ih}) \leq \frac{(1+\epsilon)area_A(i, j^*)}{area_B(i, j^*)} = (1+\epsilon)conf(i, j^*)$$
$$\leq (1+\epsilon)\hat{c}. \blacksquare$$

For running time, we state the following analogue of Theorem 1, which follows from the monotonicity of H_i^A in the balance and debit models.

THEOREM 4 *The total running time of the algorithm to select intervals for fail tableaux in the balance and debit models is $O(n \log_{1+\epsilon}(area_A(1, n)/\Delta))$ and hence $O(\frac{1}{\epsilon} n \log n)$ if $area_A(1, n)/\Delta$ is bounded by a polynomial in n.*

D. Fail Tableaux in the Credit Model

The algorithm for fail tableaux for the balance and debit models relied on monotonicity of H_i^A, which, in the credit model, equals $A_{i-1} - \min_{k \geq i}\{B_k - A_k\}$ and which is provably not monotonic. The solution is to use the breakpoints s_{il} for fail tableaux defined for the balance model! Let us define $area_A^b(i, j)$ and $area_A^c(i, j)$ to be $area_A(i, j)$ in the balance and credit model, respectively. Specifically, $area_A^b(i, j) = \sum_{l=i}^{j}[A_l - A_{i-1}]$ and $area_A^c(i, j) = \sum_{l=i}^{j}[A_l - (A_{i-1} - \min_{k \geq i}\{B_k - A_k\})]$. Define $area_B^c(i, j)$ to be (as expected) the area for B between i and j in the credit (or balance) model (specifically, $area_B^c(i, j) = \sum_{l=i}^{j}[B_l - A_{i-1}]$); define $area_B^b(i, j)$ to equal $area_B^c(i, j)$. Confidence $conf^c(i, j)$ is then defined to be $area_A^c(i, j)/area_B^c(i, j)$, if the denominator is positive. Here is the algorithm.

As in the case of fail tableaux for the balance and debit models, we again assume, *with* loss of generality, that the confidence of an optimal interval $[i, j^*]$ is positive. It is easy separately to handle the case in which $conf(i, j^*) = 0$. Once again, Δ denotes the smallest nonzero a_i or b_i.

For each $i = 1, 2, 3, ..., n$ such that $area_A^b(i, n) > 0$, do:

1) For each integer $l = 0, 1, 2, ..., \lceil \log_{1+\epsilon}(area_A^b(i, n)/\Delta) \rceil$, find the largest $j \geq i$, if one exists, such that $0 < area_A^b(i, j) \leq \Delta(1+\epsilon)^l$. Call that largest j s_{il}.
2) Compute the confidence of each interval $[i, s_{il}]$ for the same l's.
3) Output the longest interval $[i, s_{il}]$ such that $conf^c(i, s_{il}) \leq \hat{c}(1+\epsilon)$ (if such an s_{il} exists).

The next two results characterize the efficiency and correctness of the above algorithm.

THEOREM 5 *The overall running time is $O(n \log_{1+\epsilon}(area_A^b(1, n)/\Delta))$, which is $O(n \log_{1+\epsilon}(area_A^c(1, n)/\Delta))$ and hence $O(\frac{1}{\epsilon} n \log n)$ if $area_A^c(1, n)/\Delta$ is bounded by a polynomial in n.*

Proof: First we prove that $s_{i-1,l} \leq s_{il}$ for all i, l if both are defined; then we prove that the overall running time is $O(n \log_{1+\epsilon}(area_A^c(1, n)/\Delta))$.

Because s_{il} uses the balance model in its definition, that $s_{i-1,l} \leq s_{il}$ follows from the monotonicity of $H_i^A = A_{i-1}$ in the balance model.

It is obvious, from the proof of Lemma 3, that the running time is $O(n \log_{1+\epsilon}(area_A^b(1, n)/\Delta))$; what is not obvious is that the running time is $O(n \log_{1+\epsilon}(area_A^c(1, n)/\Delta))$. Yet $area_A^b(1, n) = \sum_{l=1}^{n}[A_l - A_{i-1}] \leq \sum_{l=1}^{n}[A_l - (A_{i-1} - \min_{k \geq i}\{B_k - A_k\})] = area_A^c(1, n)$. $\blacksquare$

THEOREM 6
1) *(No false positives) If the algorithm outputs an interval $[i, j]$, then $conf^c(i, j) \leq \hat{c}(1+\epsilon)$.*
2) *(No false negatives) Given i, let $j^* \geq i$ be largest such that $conf^c(i, j^*) \leq \hat{c}$ (if such a $j^* \geq i$ exists). Then the algorithm produces an interval $[i, j']$, $j' \geq j^*$, having confidence $conf^c(i, j') \leq \hat{c}(1+\epsilon)$.*

Proof: The first part is still trivial, as it is obvious from the algorithm that if the output includes an interval $[i, j]$, then $conf^c(i, j) \leq \hat{c}(1+\epsilon)$.

For the second part, define $h \geq 0$ such that $\Delta(1+\epsilon)^{h-1} < area_A^b(i, j^*) \leq \Delta(1+\epsilon)^h$ (which exists because we are assuming that $area_A^b(i, j^*)$ is positive and hence at least Δ). Since s_{ih} is the largest index j such that $area_A^b(i, j) \leq \Delta(1+\epsilon)^h$, and $area_A^b(i, j^*) \leq \Delta(1+\epsilon)^h$, it follows that $s_{ih} \geq j^*$. The algorithm did compute the confidence of interval $[i, s_{ih}]$. We now prove that $conf^c(i, s_{ih}) \leq \hat{c}(1+\epsilon)$, and hence that the algorithm will report interval $[i, s_{ih}]$ (or a longer interval, also of $conf^c$ at most $\hat{c}(1+\epsilon)$).

$$conf^c(i, s_{ih}) = \frac{area_A^c(i, s_{ih})}{area_B^c(i, s_{ih})}$$
$$= \frac{area_A^b(i, s_{ih}) + (s_{ih} - i + 1)\Delta_i}{area_B^c(i, s_{ih})}$$

(where $\Delta_i = \min_{k \geq i}\{B_k - A_k\}$)

$$\leq \frac{area_A^b(i, s_{ih}) + (s_{ih} - i + 1)\Delta_i}{area_B^c(i, j^*)}$$

(because $j^* \leq s_{ih}$).

We now prove that $area_A^b(i, s_{ih}) \leq (1+\epsilon)area_A^b(i, j^*)$ and $s_{ih} - i + 1 \leq (1+\epsilon)(j^* - i + 1)$.

First, $area_A^b(i, s_{ih}) \leq \Delta(1+\epsilon)^h$, by construction of s_{ih}, and $area_A^b(i, j^*) > \Delta(1+\epsilon)^{h-1}$. Therefore $area_A^b(i, s_{ih})/area_A^b(i, j^*) < 1+\epsilon$.

Second, because A_l is nondecreasing in l, and $j^* \leq s_{ih}$, the average value of A_l over the interval $[i, j^*]$ is at most the average value of A_l over the interval $[i, s_{ih}]$. Therefore

$$\frac{\sum_{l=i}^{j^*} A_l}{j^* - i + 1} \leq \frac{\sum_{l=i}^{s_{ih}} A_l}{s_{ih} - i + 1},$$

and hence

$$\frac{\sum_{l=i}^{j^*}[A_l - A_{i-1}]}{j^* - i + 1} \leq \frac{\sum_{l=i}^{s_{ih}}[A_l - A_{i-1}]}{s_{ih} - i + 1}.$$

From this, we have

$$\frac{area_A^b(i, j^*)}{j^* - i + 1} \leq \frac{area_A^b(i, s_{ih})}{s_{ih} - i + 1},$$

so that

$$\frac{s_{ih} - i + 1}{j^* - i + 1} \leq \frac{area_A^b(i, s_{ih})}{area_A^b(i, j^*)} < 1 + \epsilon.$$

It follows that $s_{ih} - i + 1 < (1+\epsilon)(j^* - i + 1)$. Then we have

$$\begin{aligned} conf^c(i, s_{ih}) &= area_A^c(i, s_{ih}) / area_B^c(i, s_{ih}) \\ &= \frac{[area_A^b(i, s_{ih}) + (s_{ih} - i + 1)\Delta_i]}{area_B^c(i, s_{ih})} \\ &\leq \frac{(1+\epsilon) area_A^b(i, j^*) + (1+\epsilon)(j^* - i + 1)\Delta_i}{area_B^c(i, s_{ih})} \\ &\leq \frac{(1+\epsilon) area_A^b(i, j^*) + (1+\epsilon)(j^* - i + 1)\Delta_i}{area_B^c(i, j^*)} \end{aligned}$$

$$= \frac{(1+\epsilon) area_A^c(i, j^*)}{area_B^c(i, j^*)} = (1+\epsilon) conf^c(i, j^*) \leq (1+\epsilon)\hat{c}. \blacksquare$$

IV. Experiments

We now show the utility of conservation rules in capturing data semantics and data quality problems. We also investigate the trade-off between performance and tableau quality with respect to ϵ, and we demonstrate the scalability of our algorithms. Experiments were performed on a 2.2 GHz dual-core Pentium PC with 4 GB of RAM. All the algorithms were implemented in C.

We compare our confidence metrics to two alternatives that can be efficiently evaluated by an existing interval mining algorithm [9]. The first is to sum up the counts a_i and b_i within an interval I and take the ratio. The second metric divides the area under A by the area under B within I (which corresponds to our basic model with a baseline always at zero). The idea behind this "sliding windows" approach is to consider intervals in which the sums of inbound and outbound events are quite different. These methods were instrumented using the Optimized Support Rules algorithm [9], which reports the maximal intervals from among all those having ratio outside a given value range; therefore, it considers all possible sliding window sizes. We use the following data sets:

- *NZ-Credit-Card* [19]: monthly aggregated credit card charges and payments reported by the Reserve Bank of New Zealand from 1981 to 2009; here, $n = 344$.
- *People-Count* [17]: counts of persons entering and exiting the front door of a building at the University of California, Irvine, as measured by an optical sensor ($n = 5040$ – one measurement every half an hour for 15 weeks). We also use a list of events scheduled in this building [18].
- *Network Monitoring*: counts of incoming and outgoing traffic per router for several hundred routers, collected from a large network every five minutes for roughly 2 weeks ($n = 3800$ measurements per router).
- *TCP packet traces* [20]: we use the DEC-PKT-3 trace ($n = 177802$), with a conservation law between the number of SYN packets (requests to start a TCP connection) and FIN plus RST packets (connection termination).
- *Job Log* [21]: trace of 1124772 jobs submitted to a grid computing cluster. Here the conservation law is between the number of submitted jobs and the number of completed jobs.

While the algorithms from Section III specify that Δ should be the minimum positive a_i or b_i, our implementation sets Δ equal to 1, which gives a worst-case running time conceivably larger than what is achieved by using the Δ from Section III.

A. Balance Model Example: NZ-Credit-Card data

For the most part, all bills are paid, but with a slight delay of a month. The entire sequence has a confidence close to one and is returned in the hold tableau. Figure 3 (left) shows the fail tableau with $\hat{c} = 0.8$; here, we use the balance model, which allows us to find time periods with high outstanding debt. We see more intervals from the recent years, suggesting that the total outstanding balance has risen. Also, the consistent appearance of the interval Nov-Dec suggests that people charge more than they pay during the holiday shopping season in November and December, at least between 2004 and 2007. Perhaps not surprisingly, Nov-Dec of 2008 was not reported, when credit card debt was low due to dampened consumption during an economic recession [16]. There are no tableau intervals ending in January, indicating that unpaid charges from November and December were paid by January (i.e., adding January to a low-confidence Nov-Dec interval boosts confidence above 0.8; else intervals of the form Nov-Jan would have been included in the fail tableau instead). These findings were not obvious from a plot of the instantaneous or cumulative counts.

Figure 3 (middle and right) verifies that December charges were higher than December payments, but January payments were higher than January charges; it shows the charges and payments made *only* in Decembers (resp., Januaries) of each year. In the December plot, charges dominate payments, especially in the last five years. In the January plot, payments dominate charges.

In contrast, Optimized Support Rules [9], run over a variety of confidence thresholds, did not discover these patterns. Summing up the a's and b's within each interval, the fail tableau with $\hat{c} = 0.8$ contains a single interval of length one (Jan 1981). Since the magnitudes of monthly charges and payments have increased over time, this result suggests that the difference between charges and payments was proportionately larger in 1981. With $\hat{c} = 0.9$, we get only the three small intervals: Jan-Mar 1981, Dec 2003 and Dec 2008. If, instead, we use cumulative amounts, the fail tableau contains only Jan-Feb 1981 with $\hat{c} = 0.8$, and Jan-May 1981 with $\hat{c} = 0.9$. To explain this, observe that taking the ratio of the area under A and the area under B within an interval amounts to using a baseline of zero. Thus, intervals starting later in the sequence end up with artificially high confidences that are well above $\hat{c}$, and therefore are not selected for the fail tableau.

B. Credit Model Example: People-Count data

Figure 4 shows a fragment of the cumulative entrance and exit counts for this data set. There is a persistent violation of the conservation law, perhaps due to an unmonitored side

Month	Year
Nov-Dec	2007
July-Aug	2007
Nov-Dec	2006
Oct-Nov	2006
May-June	2006
Nov-Dec	2005
Aug-Sept	2005
Nov-Dec	2004

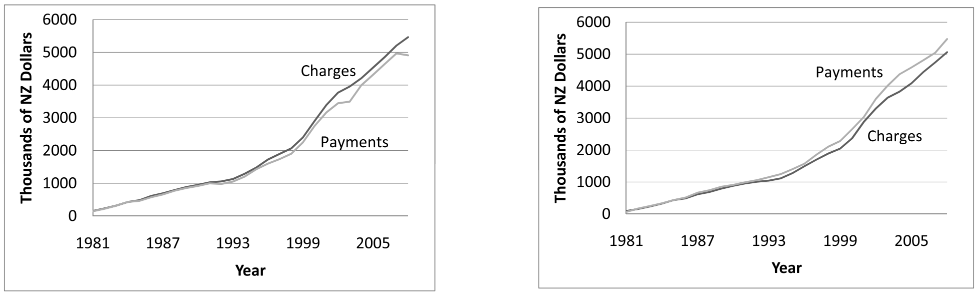


Fig. 3. Fail tableau for NZ-Credit-Card data using a balance model and $\hat{c} = 0.8$ (left), charges and payments in New Zealand in Decembers between 1981-2008 (middle), charges and payments in New Zealand in Januaries between 1981-2008 (right)

TABLE I
SELECTED EVENTS (LEFT) AND CORRESPONDING FAIL TABLEAU INTERVALS FROM THE SAME DAY (RIGHT) IN PEOPLE-COUNT DATA

Event date and time	Tableau interval(s) from the same day
August 2, 15:30-16:30	15:30-16:00
August 4, 16:30-17:30	12:30-14:30, 16:00-17:00, 16:30-17:30
August 9, 08:00-16:00 August 9, 11:00-14:00	06:30-17:00
August 12, 08:00-11:00	06:30-12:00
August 18, 08:00-17:00	06:00-13:30, 12:00-16:30, 13:30-17:30
August 18, 18:00-20:30	17:00-19:00, 17:30-20:30

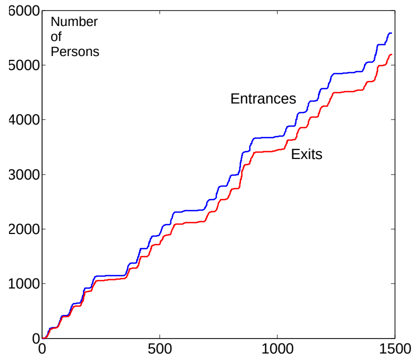


Fig. 4. Cumulative entrance and exit counts for the People-Count data set

exit, and possibly some local violations due to delays. As a result, when using the balance model, only short intervals from the beginning of the sequence appear in the hold tableaux; in the fail tableaux, all later intervals of the sequence have low confidence due to accrued imbalances. Instead, we use the credit model to account for potentially unmonitored exits, by crediting all "leftover" people in the building at the beginning of an interval as having exited.

During August, fourteen known events were scheduled in this building. In the time intervals surrounding these events, we expect a temporary discrepancy between the number of people entering the building and the number of people leaving. We generated a fail tableau with $\hat{c} = 0.6$ and found that they correspond to these known events. Note that since we are using the credit model, any reported intervals have low confidence due to divergence (i.e., delays between entrance and exit times) within them and not due to imbalances carried over from the past. In Table I, we report the maximal intervals corresponding to five days in August during which at least one known event took place. Our intervals closely match the known events, whose durations are shown on the left. On August 4th, in addition to the two intervals spanning the event, we also report the interval 12:30-14:30; this is likely because of the imbalance of traffic occurring during lunch time. To validate this, we examined the maximal intervals on other days in this data set, when no event was scheduled, and found that either no intervals were returned, or some intervals in between 11:30 and 15:00 were obtained. Once again, these findings are clearly not obvious from the instantaneous event counts or the derived cumulative counts.

TABLE II
FAIL TABLEAU FOR THE NETWORK MONITORING DATA SET

Router name	Interval
Router-1	1 - 3800
Router-10	1 - 3800
Router-12	1 - 3800
Router-6	1 - 3800
Router-25	1 - 3800
Router-7	1 - 3610

Using Optimized Support Rules on instantaneous data with various confidence thresholds, we obtained fail tableaux whose intervals did not overlap with known scheduled events, with only a few exceptions. This is because delays cannot be accurately modeled by adding up entrances and exits within an interval. Some of the reported intervals extended into the following day and almost all days included intervals at odd hours of the day. Using cumulative counts yielded intervals at the beginning of the sequences, which were not meaningful.

C. *Debit Model Example: Network Monitoring data*

We now examine network monitoring data to find cases in which some links of a router are not monitored, causing a discrepancy between the measured inbound and outbound traffic. To ignore accrued imbalances, we use the debit model which subtracts prior incoming packets without outgoing counterparts. This data set contains several hundred pairs of sequences, one for each router. Table II shows the fail tableau intervals with $\hat{c} = 0.5$, along with the router names. These suggest that one or more links on Router-7 were not being

TABLE III
HOLD TABLEAUX FOR ROUTER-7

confidence above 0.99
3650 - 3660
3660 - 3680
3790 - 3800

confidence above 0.9
3530 - 3800

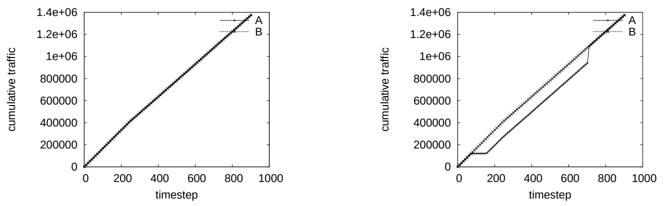


Fig. 5. Well-behaved (left) and perturbed (right) network data

monitored up to around time 3610 and started being monitored afterward. To confirm this, we show two *hold* tableaux for this router in Table III. Interestingly, only three short intervals have confidence above 0.99, suggesting that even if all links are monitored correctly, small violations of the conservation law are normal. Using $\hat{c} = 0.9$ yields a longer interval that only slightly overlaps with the "bad" interval from the fail tableau.

Interestingly, Optimized Support Rules gave very similar results on this data set because all the conservation law violations are due to loss (of data flowing on unmonitored links), not delay. However, if there had also been delays (e.g., if some links were to report their traffic late), then we suspect that only our techniques would have been able to detect them.

D. *Perturbed Data*

The previous experiments have demonstrated that the generated intervals correspond to some event in the data, i.e., no false positives. Since we lack "ground truth" of the complete set of such events in these data sets, here we complement those experiments by applying CRs to artificially perturbed data, to verify that our techniques do not miss anything "interesting", that is, there are no false negatives.

We started with a "well behaved" real data set, containing 906 incoming and outgoing traffic measurements. The confidence of the data in the entire interval $[1, 906]$, using the balance model, was nearly one, and we obtained empty fail tableaux with a confidence bound as high as 0.3. Thus, there were no reportable events in this data set. We then introduced controlled amounts of delay to the outgoing traffic a as follows. We removed a fraction of total traffic d at the time step i with highest a_i, such that the cumulative amount $d\sum_{\ell=1}^{n} a_\ell$ was subtracted from consecutive elements $a_i, a_{i+1}, ..., a_j$, subject to these values' not falling below 0. Then, at some random index $i' > i$, the previously subtracted quantity $d\sum_{\ell=1}^{n} a_\ell$ was added to $a_{i'}$ to compensate. In our experiments, we used $d \in \{0.01, 0.1, 0.25\}$; Figure 5 shows the cumulative curves with $d = 0.1$ (right) in comparison to those with $d = 0$ (left).

The hold tableau with $\hat{c}$=0.99 picked up two intervals which correspond almost exactly to the period before the drop was introduced and the period after the drop was compensated for, respectively. The fail tableau with $\hat{c}$=0.1 reported (roughly) the interval containing the drop, as well as very short intervals until the drop was compensated for, indicating that there is an initial imbalance but no further problem compounding this imbalance. This is exactly what one would expect from the balance model. When there was loss rather than delay, hold tableaux indeed picked up only the interval before the loss, and fail tableaux picked up intervals until the end of time.

In addition to varying the magnitude of temporary loss, we looked at the effect of how gradually it occurs. We dampened the drop of traffic to at most 25% (rather than allowing a complete drop to 0 at each time step). Hold tableaux again resulted in two intervals, corresponding to the periods before loss and after recovery, respectively, though with less tightly fitting intervals.

With delay, the credit and debit models were unable to discount much history due to the minimum distance between the curves' being small and, therefore, gave similar results to the balance model. With loss, hold tableaux at high confidence reported an interval before the sudden drop and an interval after the drop period to the end. As expected, fail tableaux only reported the period of traffic drop, with no intervals at subsequent times, due to crediting/debiting this drop.

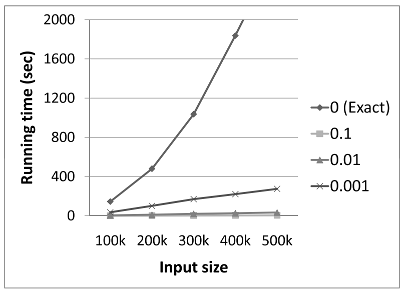
The approximate algorithm with $\epsilon = 0.01$ generated almost identical intervals to that using the exact algorithm for hold tableaux, and less than 10% longer intervals for fail tableaux. Otherwise, all the reported results above were the same. With $\epsilon = 0.1$, more intervals were generated (and those that were generated were more loosely fitting), with clusters of overlapping intervals. However, each interval generated by the exact algorithm corresponded to a single cluster.

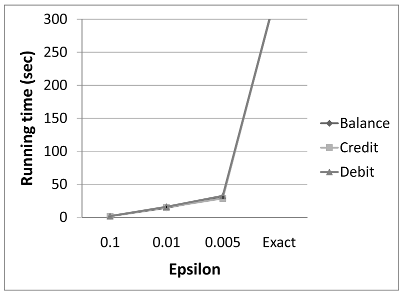
Finally, we compared against the results obtained from using Optimized Support Rules [9]. Using the instantaneous curves, this method only identified the traffic loss in $[74, 155]$ for all cases. Using the cumulative curves, the period of loss was not detected immediately as it occurred, but several intervals were reported sometime between the loss and recovery. Furthermore, it could not distinguish between delay and loss.

In general, using tighter confidence values gave more precise characterization of delay or loss, but sometimes came at the price of false-negatives. Therefore, selecting a value for $\hat{c}$ should depend on the relative importance of preventing false-negatives and pinpointing true-positives.

E. *Performance and Scalability*

Figure 6 compares the running time of our algorithms against a quadratic-time algorithm that considers every possible interval. We only report the wall clock time for generating candidate maximal intervals for a hold tableau, and exclude the preprocessing and tableau construction times, which are both linear. Time is measured in seconds using the Unix `clock` command. On the left, we plot the performance for several values of ϵ on various prefixes of Job-Log data, using the balance model and a $\hat{c}$ slightly higher than the confidence value of the entire data. Our algorithm scales gracefully, and,





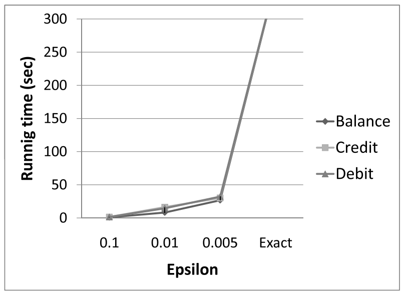


Fig. 6. Running time on the job-log data (left) and DEC-PKT-3 data (hold tableau–middle, fail tableau–right)

even for small values of ϵ such as 0.001, is an order of magnitude faster. The middle figure plots the running times to generate hold tableau intervals using DEC-PKT-3 data for various confidence models and values of ϵ; the figure on the right repeats this for fail tableaux. Again, we use a confidence threshold that is slightly higher than the confidence of the overall data set. As before, we observe an order-of-magnitude performance improvement, even for small ϵ.

V. Improved Algorithms for the Balance Model

In Section III, we gave algorithms to generate candidate intervals for the balance, credit and debit models that run in $O(\frac{1}{\epsilon}n\log_2(area_B(1,n)/\Delta))$ time for hold tableaux, and in $O(\frac{1}{\epsilon}n\log_2(area_A(1,n)/\Delta))$ time for fail tableaux. Recall that we are given two nonnegative real sequences, $a = \langle a_1, a_2, \ldots, a_n\rangle$ and $b = \langle b_1, b_2, \ldots, b_n\rangle$, and Δ is defined to be the minimum positive a_i or b_i, over $i = 1, 2, \ldots, n$. From a and b, the cumulative time series A and B can be obtained in linear time. Also, by simple linear-time preprocessing, the confidence of any interval can be computed in constant time. To summarize, $area_A(i,j) = \sum_{l=i}^{j}(A_l - H_i^A)$ and $area_B(i,j) = \sum_{l=i}^{j}(B_l - H_i^B)$. The confidence $conf(i,j)$ equals $area_A(i,j)/area_B(i,j)$, when the denominator is greater than zero. The sequences $\langle H_1^A, H_2^A, H_3^A, \ldots, H_n^A\rangle$ and $\langle H_1^B, H_2^B, H_3^B, \ldots, H_n^B\rangle$ are defined in a model-dependent way. For example, in the balance model, $H_i^A = H_i^B = A_{i-1}$.

If $area_B(1,n)/\Delta$ is bounded by a polynomial in n, then the running time to generate the candidate maximal intervals is $O(\frac{1}{\epsilon}n\log n)$. In principle, $area_B(1,n)/\Delta$ can be an exponential function of n, and as such, the worst case running time can become $O(n^2)$; but we have argued that if $area_B(1,n)/\Delta$ is exponential in n, then it would take linear memory even to store *one* of the numbers exactly, and then doing arithmetic on such long numbers would be prohibitively slow. In all the data sets we considered, we have indeed found the running time to be much better than quadratic.

Nonetheless, it is possible for the area to grow superlinearly and, as such, the constants hidden in the big O-notation may be nonnegligible. An algorithm that has time complexity independent of area and runs for $O(\frac{1}{\epsilon}n\log n)$ steps is interesting not just from a theoretical perspective, but could achieve better running time in practice.

In this section, we give such algorithms for the balance model, for both hold and fail tableaux. Ignoring the preprocessing steps to compute the cumulative sequences and the constant time to compute the confidence of each interval, the algorithms run in $O(n\log_{1+\epsilon} n)$ time, which is $O(\frac{1}{\epsilon}n\log n)$ when $\epsilon \leq 1$.

A. General Framework

Candidate intervals can be generated by considering, for each i, all linearly many intervals $[i,j]$. We then compute the confidence for each chosen interval individually to check if it satisfies the threshold $\hat{c}$, and return the largest j such that $[i,j]$ satisfies confidence. The time complexity of this naive exhaustive search is $\Theta(n^2)$. In Section III, in order to overcome the barrier of quadratic running time, we trade off accuracy for speed. We generate for each i, a sparse set of intervals $[i,j]$ and output the longest of the selected intervals having confidence at least $\hat{c}/(1+\epsilon)$, if any exists. Even though the algorithm tests only a sparse set of intervals starting at i, we can guarantee that there is no false positive, that is, the returned intervals have confidence at least $\hat{c}/(1+\epsilon)$. In addition, if there exists an interval starting at i with confidence $\hat{c}$, and if $[i,j^*]$ be the largest among those, then the algorithm returns an interval $[i,j']$, $j' \geq j^*$ with confidence at least $\hat{c}/(1+\epsilon)$. This implies there are no false negatives. To create the sparse set of intervals, the algorithm selects the endpoints based on geometric growth of area under the curves A and B. This brings in the dependence on area in the running time, as the number of intervals that are checked for each i is dependent on the rate of area growth.

One possible way to remove the dependence on area is to generate the sparse set of intervals based on *lengths* of intervals. However, as has already been argued, since the curves can grow an unbounded amount between time steps, there is no guarantee that the confidence of $[i,j]$ will be at least $\hat{c}/(1+\epsilon)$. In light of that, it seems improbable that the general framework of generating a sparse set of intervals can simultaneously be made to have time complexity independent of area and provide guarantees of not having any false positives or false negatives. Surprisingly, we show indeed both of these can be achieved. Our algorithms are based on the following observations:

1) There is an alternate way to generate candidate maximal intervals. We can consider each endpoint j, instead of start point, and return for each endpoint j the smallest i such that $[i,j]$ satisfies confidence.

2) For each endpoint j, instead of considering all linearly many possible start points, we can consider interval starting points based on geometric growth in length. Fixing right endpoints instead of left ones enables us to argue that, between adjacent start points, even though the area growth can be significant, it does not affect the guarantees.

Next, we describe the algorithms for hold and fail tableaux in detail.

B. *Hold Tableau Interval Selection*

The algorithm for hold tableaux for the balance model is as follows.

For $j = 1, 2, 3, ..., n$, do:

1) For each integer $h = 0, 1, 2, ..., \lceil \log_{1+\epsilon}(j) \rceil$, find the smallest integer $i \leq j$, if one exists, such that $j - i + 1 \leq (1+\epsilon)^h$. Call that smallest i, l_{jh}.
2) Compute the confidence of each interval $[l_{jh}, j]$ for $h = 0, 1, 2, ..., \lceil \log_{1+\epsilon}(j) \rceil$.
3) Output the longest interval $[l_{jh}, j]$ of confidence at least $\hat{c}/(1+\epsilon)$, if at least one exists.

THEOREM 7 *The total running time of the given algorithm for the balance model is $O(n \log_{1+\epsilon} n)$, which is $O(\frac{1}{\epsilon} n \log_2(n))$ if $\epsilon \leq 1$.*

Proof: First, computing $area_A(i,j)$, $area_B(i,j)$, and $conf(i,j)$ are constant-time operations, since we can precompute $S_j = \sum_{l=1}^{j} A_l$ and $T_j = \sum_{l=1}^{j} B_l$. Then
$area_A(i,j) = (S_j - S_{i-1}) - (j-i+1)H_i^A$,
$area_B(i,j) = (T_j - T_{i-1}) - (j-i+1)H_i^B$, and
$conf(i,j) = area_A(i,j)/area_B(i,j)$,
provided the denominator is positive.

For each endpoint j, we compute $1 + \lceil \log_{1+\epsilon}(j) \rceil$ starting points. Each of these starting points can be computed in constant time. Hence the total time to compute all the intervals for which the confidence is tested is $\sum_{j=1}^{n} (1 + \lceil \log_{1+\epsilon}(j) \rceil)$, which is $O(n \log_{1+\epsilon} n)$. ■

The next theorem states that the intervals produced by the algorithm are accurate.

THEOREM 8 1) *(No false positives) If the algorithm outputs an interval $[i,j]$, then $conf(i,j) \geq \hat{c}/(1+\epsilon)$.*
2) *(No false negatives) Given j, let $i^* \leq j$ be smallest such that $conf(i^*,j) \geq \hat{c}$ (if such an $i^* \leq j$ exists). Then the algorithm produces an interval $[i',j]$, $i' \leq i^*$, having confidence at least $\hat{c}/(1+\epsilon)$.*

Proof: The first part is trivial, as it is obvious from the last step of the algorithm that if the output includes an interval $[i,j]$, then $conf(i,j) \geq \hat{c}/(1+\epsilon)$. Modulo the distinction between $\hat{c}$ and $\hat{c}/(1+\epsilon)$, there are no "false positives."

For the second part, define k such that $(1+\epsilon)^{k-1} < j - i^* + 1 \leq (1+\epsilon)^k$. Since l_{jk} is the smallest index i such that $j - i + 1 \leq (1+\epsilon)^k$, and $j - i^* + 1 \leq (1+\epsilon)^k$, it follows that $l_{jk} \leq i^*$. The algorithm did compute the confidence of interval $[l_{jk}, j]$. We now prove that $conf(l_{jk}, j) \geq \hat{c}/(1+\epsilon)$, and hence that the algorithm will report interval $[l_{jk}, j]$ (or a longer interval, also of confidence at least $\hat{c}/(1+\epsilon)$).

$$conf(l_{jk}, j) = \frac{area_A(l_{jk}, j)}{area_B(l_{jk}, j)}$$
$$= \frac{\sum_{r=l_{jk}}^{j} (A_r - A_{l_{jk}-1})}{\sum_{r=l_{jk}}^{j} (B_r - A_{l_{jk}-1})}.$$

We can rewrite

$$\begin{aligned} & area_A(l_{jk}, j) \\ = \; & \sum_{r=l_{jk}}^{i^*-1} (A_r - A_{l_{jk}-1}) + \sum_{r=i^*}^{j} (A_r - A_{l_{jk}-1}) \\ = \; & area_A(l_{jk}, i^*-1) + \sum_{r=i^*}^{j} (A_r - A_{i^*-1}) \\ & + (j - i^* + 1)(A_{i^*-1} - A_{l_{jk}-1}) \\ = \; & area_A(l_{jk}, i^*-1) + area_A(i^*, j) \\ & + (j - i^* + 1)(A_{i^*-1} - A_{l_{jk}-1}). \end{aligned}$$

Similarly, opening up the expression for $area_B(l_{jk}, j)$, we get

$$area_B(l_{jk}, i^*-1) + area_B(i^*, j)$$
$$+ (j - i^* + 1)(A_{i^*-1} - A_{l_{jk}-1}).$$

Now since B is a monotonically nondecreasing sequence, we have

$$area_B(l_{jk}, i^*-1) \leq (i^* - l_{jk})(B_{i^*-1} - A_{l_{jk}-1})$$
$$= (i^* - l_{jk})(B_{i^*-1} - A_{i^*-1})$$
$$+ (i^* - l_{jk})(A_{i^*-1} - A_{l_{jk}-1}).$$

We infer from the choice of l_{jk} that $j - l_{jk} + 1 = \lfloor (1+\epsilon)^k \rfloor$. We also have $(1+\epsilon)^{k-1} < j - i^* + 1 \leq (1+\epsilon)^k$. Hence $j - l_{jk} + 1 \leq (1+\epsilon)(j - i^* + 1)$. Equivalently, $(j - i^* + 1) + (i^* - l_{jk}) \leq (1+\epsilon)(j - i^* + 1)$. This implies that $i^* - l_{jk} \leq \epsilon(j - i^* + 1)$. Again by monotonicity of B, $area_B(i^*, j) \geq (j - i^* + 1)(B_{i^*-1} - A_{i^*-1})$. This implies that

$$area_B(l_{jk}, i^*-1)$$
$$= (i^* - l_{jk})(B_{i^*-1} - A_{i^*-1})$$
$$+ (i^* - l_{jk})(A_{i^*-1} - A_{l_{jk}-1})$$
$$\leq \epsilon(j - i^* + 1)(B_{i^*-1} - A_{i^*-1})$$
$$+ \epsilon(j - i^* + 1)(A_{i^*-1} - A_{l_{jk}-1})$$
$$\leq \epsilon[area_B(i^*, j) + (j - i^* + 1)(A_{i^*-1} - A_{l_{jk}-1})].$$

Noting that $area_A(l_{jk}, i^*-1) \geq 0$, and letting $X = (j - i^* + 1)(A_{i^*-1} - A_{l_{jk}-1})$, we get

$$conf(l_{jk}, j) = \frac{area_A(l_{jk}, j)}{area_B(l_{jk}, j)}$$
$$= \frac{area_A(l_{jk}, i^*-1) + area_A(i^*, j) + X}{area_B(l_{jk}, i^*-1) + area_B(i^*, j) + X}$$

$$\geq \frac{area_A(i^*, j) + X}{(1+\epsilon)[area_B(i^*, j) + X]}.$$

Using simple algebra and using the fact that $area_B(i^*, j) \geq area_A(i^*, j)$, we get

$$(1+\epsilon)conf(l_{jk}, j) \geq \frac{area_A(i^*, j) + X}{area_B(i^*, j) + X}$$

$$\geq \frac{area_A(i^*, j)}{area_B(i^*, j)} = conf(i^*, j) \geq \hat{c}.$$

Therefore, we have $conf(l_{jk}, j) \geq \hat{c}/(1+\epsilon)$. ■

C. *Fail Tableau Interval Selection*

The algorithm for fail tableaux for the balance model is as follows.

For $j = 1, 2, 3, ..., n$, do:

1) For each integer $h = 0, 1, 2, ..., \lceil \log_{1+\epsilon}(j) \rceil$, find the smallest integer $i \leq j$, if one exists, such that $j - i + 1 \leq (1+\epsilon)^h$. Call that smallest i, m_{jh}.
2) Compute the confidence of each interval $[m_{jh}, j]$ for $h = 0, 1, 2, ..., \lceil \log_{1+\epsilon}(j) \rceil$.
3) Output the longest interval $[m_{jh}, j]$ of confidence at most $\hat{c}(1+\epsilon)$, if at least one exists.

The running time is clearly the same as that of the algorithm for hold tableau interval selection given in Theorem 7.

The next theorem states that the intervals produced by the algorithm are accurate.

THEOREM 9 1) *(No false positives) If the algorithm outputs an interval* $[i, j]$, *then* $conf(i, j) \leq \hat{c}(1+\epsilon)$.
2) *(No false negatives) Given* j, *let* $i^* \leq j$ *be smallest such that* $conf(i^*, j) \leq \hat{c}$ *(if such an* $i^* \leq j$ *exists). Then the algorithm produces an interval* $[i', j]$, $j - i' + 1 \geq (j - i^* + 1)/(1+\epsilon)$, *having confidence at most* $\hat{c}(1+\epsilon)$.

Proof: The first part is trivial, as it is obvious from the last step of the algorithm that if the output includes an interval $[i, j]$, then $conf(i, j) \leq \hat{c}(1+\epsilon)$.

For the second part, define k such that $\lfloor (1+\epsilon)^k \rfloor \leq j - i^* + 1 < \lfloor (1+\epsilon)^k \rfloor (1+\epsilon)$. Since m_{jk} is the smallest index i such that $j - i + 1 \leq (1+\epsilon)^k$, $j - m_{jk} + 1 = \lfloor (1+\epsilon)^k \rfloor$. From $j - i^* + 1 \leq \lfloor (1+\epsilon)^k \rfloor (1+\epsilon)$, it follows that $j - m_{jk} + 1 \geq (j - i^* + 1)/(1+\epsilon)$. The algorithm did compute the confidence of interval $[m_{jk}, j]$. We now prove that $conf(m_{jk}, j) \leq \hat{c}(1+\epsilon)$, and hence that the algorithm will report interval $[m_{jk}, j]$ (or a longer interval, also of confidence at most $\hat{c}(1+\epsilon)$).

We prove this by contradiction. Suppose that

$$conf(m_{jk}, j) = \frac{area_A(m_{jk}, j)}{area_B(m_{jk}, j)} > \hat{c}(1+\epsilon).$$

We have

$$conf(i^*, j) = \frac{area_A(i^*, j)}{area_B(i^*, j)} = \frac{\sum_{r=i^*}^{j} (A_r - A_{i^*-1})}{\sum_{r=i^*}^{j} (B_r - A_{i^*-1})}.$$

We can rewrite $area_A(i^*, j)$ as

$$area_A(i^*, m_{jk} - 1) + area_A(m_{jk}, j)$$

$$+(j - m_{jk} + 1)(A_{m_{jk}-1} - A_{i^*-1}).$$

Similarly, opening up the expression for $area_B(i^*, j)$, we get

$$area_B(i^*, m_{jk} - 1) + area_B(m_{jk}, j)$$

$$+(j - m_{jk} + 1)(A_{m_{jk}-1} - A_{i^*-1}).$$

Now since B is a monotonically nondecreasing sequence, we have

$$area_B(i^*, m_{jk} - 1)$$

$$\leq (m_{jk} - i^*)(B_{m_{jk}-1} - A_{i^*-1})$$

$$= (m_{jk} - i^*)(B_{m_{jk}-1} - A_{m_{jk}-1})$$

$$+(m_{jk} - i^*)(A_{m_{jk}-1} - A_{i^*-1}).$$

From the choice of m_{jk}, $j - m_{jk} + 1 = \lfloor (1+\epsilon)^k \rfloor$ and we have $\lfloor (1+\epsilon)^k \rfloor \leq j - i^* + 1 < \lfloor (1+\epsilon)^k \rfloor (1+\epsilon)$. Hence $j - i^* + 1 \leq (1+\epsilon)(j - m_{jk} + 1)$. Equivalently, $(j - m_{jk} + 1) + (m_{jk} - i^*) \leq (1+\epsilon)(j - m_{jk} + 1)$. This implies that $m_{jk} - i^* \leq \epsilon(j - m_{jk} + 1)$. Again by monotonicity of B, $area_B(m_{jk}, j) \geq (j - m_{jk} + 1)(B_{m_{jk}-1} - A_{m_{jk}-1})$. This implies that

$$area_B(i^*, m_{jk} - 1)$$

$$= (m_{jk} - i^*)(B_{m_{jk}-1} - A_{m_{jk}-1})$$

$$+(m_{jk} - i^*)(A_{m_{jk}-1} - A_{i^*-1})$$

$$\leq \epsilon(j - m_{jk} + 1)(B_{m_{jk}-1} - A_{m_{jk}-1})$$

$$+\epsilon(j - m_{jk} + 1)(A_{m_{jk}-1} - A_{i^*-1})$$

$$\leq \epsilon[area_B(m_{jk}, j) + (j - m_{jk} + 1)(A_{m_{jk}-1} - A_{i^*-1})].$$

Noting that $area_A(i^*, m_{jk} - 1) \geq 0$, and letting $Y = (j - m_{jk} + 1)(A_{m_{jk}-1} - A_{i^*-1})$, we get

$$conf(i^*, j) = \frac{area_A(i^*, j)}{area_B(i^*, j)}$$

$$= \frac{area_A(i^*, m_{jk} - 1) + area_A(m_{jk}, j) + Y}{area_B(i^*, m_{jk} - 1) + area_B(m_{jk}, j) + Y}$$

$$\geq \frac{area_A(m_{jk}, j) + Y}{(1+\epsilon)(area_B(m_{jk}, j) + Y)}$$

$$\geq \frac{area_A(m_{jk}, j)}{(1+\epsilon)area_B(m_{jk}, j)} > \hat{c}.$$

This contradicts the fact that $conf(i^*, j) \leq \hat{c}$. Therefore, we must have $conf(m_{jk}, j) \leq \hat{c}(1+\epsilon)$. ■

VI. Experimental Comparisons of Different Algorithms for the Balance Model

Since the area-based and non area-based algorithms test a different set of intervals (the former anchored on every left endpoint, the latter on every right endpoint), it is possible that their respective result sets may differ. In our experiments, we did not actually see much difference. (See Section IV for experimental setup and descriptions of data sets.) Using the NZ-Credit-Card data, we obtained exactly the same set of intervals shown in Figure 3 with $\epsilon = 0.01$, for both the area-based and non area-based algorithms. Using the DEC-PKT-3 data, most of the intervals were exactly the same. The other reported intervals had considerably overlapping counterparts between the two algorithms, with area-based reporting these intervals starting at smaller i's whereas non area-based tended to find them starting at larger i's due to the different manners of testing. The remaining experiments focus on algorithm performance.

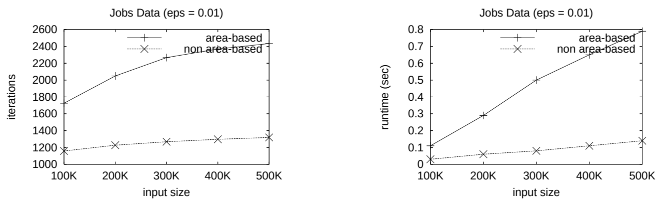


Fig. 7. Comparison of Area-Based and Non Area-Based (Hold Intervals)

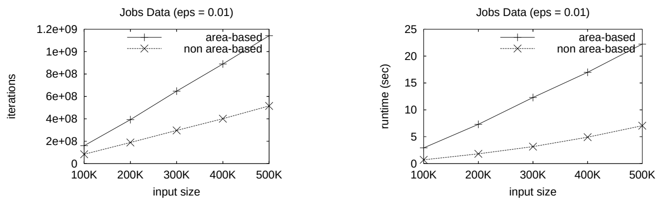


Fig. 8. Comparison of Area-Based and Non Area-Based (Fail Intervals)

First, we compared the area-based and non area-based algorithms for the balance model, implemented as they are described in Sections III and V, respectively. Using the Job-Log data, Figure 7 (left) plots the total number of intervals tested and Figure 7 (right) plots the running time in seconds, as a function of n (with $\epsilon = 0.01$), for the case of hold intervals with $\hat{c} = 0.99999$. Since the interval $[1, n]$ has very high confidence, for all the values of n from 100K to 500K, the threshold $0.99999/(1+\epsilon)$ allowed $[1, n]$ to be selected; therefore, the algorithm only tested intervals anchored at left endpoint $i = 1$ for the area-based algorithm, and at right endpoint j for the non area-based algorithm. The algorithm analysis predicts that the gap could be roughly $\log(area_B(1,n))/\log n$, and indeed the number of interval tests follows this ratio very closely, which at $n = 100$K is 1.49 (versus 1.55 observed) and at $n = 500$K is 1.84 (versus 1.9). However, the gap in running time appears to grow at a faster rate than this (from a factor of 3.67 to 5.64), perhaps due to the overhead of the more complicated area-based algorithm. Interestingly, we observed that the running times of the methods diverged even faster at smaller ϵ, despite the ratio of the number of interval tests being independent of ϵ. Figure 8 shows similar graphs for the case of fail intervals ($\hat{c} = 0.1$). Here less than $n/10$ intervals were selected, mostly (possibly overlapping) intervals of size 50-200, plus many of size 1 having confidence 0. Unlike for hold intervals, the area-based algorithm wound up testing many more intervals, anchored at all left endpoints from 1 to n; the non area-based algorithm tested intervals anchored at all right endpoints from n down to 1. Here the number of interval tests does not appear to taper off like it did in Figure 7 (left). This is because a more accurate estimate of interval tests for the area-based algorithm in this case is $\sum_{i=1}^{n} \log(area_B(i,n))/\sum_{i=1}^{n}\log(n-i+1)$.

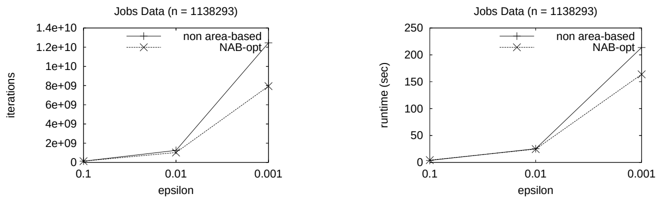


Fig. 9. Effect of Improved Interval Testing on NAB (Fail Intervals)

Notice that, when ϵ and h are small, much effort is wasted in testing intervals of length $(1+\epsilon)^h$ that have the same value of $\lfloor (1+\epsilon)^h \rfloor$. For example, when $\epsilon = 0.1$, the length of $(1+\epsilon)^h$ does not grow to more than 2 until $h = 8$, and then does not exceed 3 until $h = 12$. In general, the quantity $(1+\epsilon)^h - (1+\epsilon)^{h-1}$ does not grow by at least 1 until $h = O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$. The non area-based algorithm incurs this overhead, for each right endpoint j, by insisting that the intervals be of absolute lengths $\ell = (1+\epsilon)^h$, for some h; however, this overhead can be avoided by simply setting ℓ recursively based on the length previously tested via $\ell = \min(\ell+1, \lfloor (1+\epsilon)\ell \rfloor)$. Figure 9 compares the original NAB algorithm with this improvement, as a function of ϵ (with $n = 1,138,293$), in terms of both number of interval tests and running time, for the fail case. Clearly, this technique improves performance, as shown by the increasing gap in both interval tests and running time as ϵ decreases. Similar trends were obtained using DEC-PKT-3.

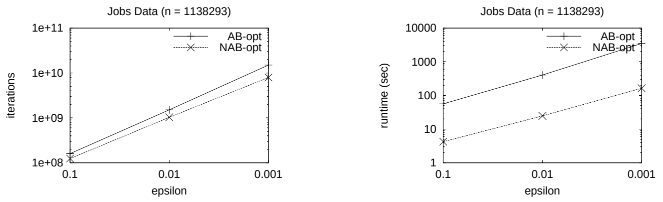


Fig. 10. Comparison of Improved Area-Based and Non Area-Based (Fail Intervals)

Unfortunately, the analog of this simple improvement is not readily implementable in the area-based algorithm due to the way the pointers at the r_{il}-positions are managed. Recall that, in order to achieve the running time described in Lemma

3, absolute area growths must be used for determining the right points along the B curve for hold intervals (and A curve for fail intervals). That is, these pointers cannot be assigned positions relative to their predecessors and cannot be safely spread out to distinct positions without potentially violating the accuracy guarantees. Therefore, we used binary search to find the r_{il}'s from scratch, for each left endpoint i; this allows us to ensure that $r_{il} > r_{i,l-1}$ and that $area_B(i, r_{il})/area_B(i, r_{i,l-1})$ is as close as possible to $(1+\epsilon)$. Figure 10 compares the two revised versions of the algorithms in terms of intervals tested and running time in log-scale, for the case of fail intervals. The graphs show that the area-based algorithm also benefits by testing fewer intervals. However, since binary search is needed to employ this technique, the running time of the area-based algorithm is worse by more than an order of magnitude.

There are further optimizations that can be done to improve the running time. For example, the order in which intervals are tested does not affect the final result. Therefore, if we test interval lengths from largest to smallest, we can stop when the first interval satisfying confidence is found, since any additional intervals satisfying confidence will be subsumed. For exposition, these were not implemented in the experiments presented.

VII. Related Work

Concepts similar to conservation laws have been discussed in network traffic analysis [10], [14], clustering data for deduplication [3], and consistent queries with aggregation constraints [6]. However, this paper is the first to address confidence metrics and tableau discovery for rules that assert such laws. Moreover, [1] studies the problem of recovering individual traffic flows from aggregated counts, which could benefit from employing conservation rules to verify the accuracy of the sums upon which such inference is based.

The data mining literature investigated finding intervals that satisfy various properties, e.g., finding intervals in an array having a mean value above a specified minimum [9]. As we showed, the confidence metrics supported by the Optimized Support Rules algorithm from [9] are not as effective as ours in the context of conservation laws (the technical reason why our confidence metrics are not compatible with this algorithm is that it effectively requires a single fixed baseline for all intervals).

There has been a great deal of work on mining time sequences [5], including similarity search and generating summaries [13]. However, our confidence measure is different from that used for time series similarity. To achieve high confidence with respect to a conservation rule, the two cumulative curves must track each other closely, but they need not be similar in terms of pattern- and shape-based properties commonly used for matching (e.g., translation and scale invariance, warping). Conversely, two time series considered similar could violate a conservation rule. Moreover, our goal is not to summarize a sequence, but rather to summarize the behavior of a conservation law between two related sequences.

There has also been prior work on event and anomaly detection [15]. Our work is orthogonal; in Section 4.1, we discussed detecting events that occurred inside a building only to validate our conservation rules.

Finally, this paper is related to work on tableau discovery for functional [4], [8], [11] and sequential dependencies [12]. However, neither of these constraints (or any others proposed in the literature) asserts any properties over counts as they are aggregated in some specified order. While we borrow the general idea of tableau discovery, we propose novel constraints and confidence metrics, and novel algorithms for efficiently identifying candidate intervals. In particular, the interval finding algorithm from [12] cannot be adapted to CRs since it crucially relies on the property that the confidences of two highly overlapping intervals of similar sizes must be similar. This is not true with CRs. To see this, take any interval and add a single arbitrarily large b_i with a corresponding $a_i = 0$.

VIII. Conclusions

In this paper, we proposed rules that express conservation laws between related quantities, such as those between the inbound and outbound counts reported by network monitoring systems. We presented several confidence metrics for conservation rules, and we gave efficient approximation algorithms for the tableau discovery problem, i.e., finding a concise set of intervals that satisfy (or fail) a supplied conservation rule given a confidence threshold. Using real data sets, we demonstrated the utility of tableau discovery for conservation rules and the efficiency of our algorithms. The reported tableaux are concise, easy to understand, and suggest interesting subsets of the data for further analysis.

References

[1] E. Airoldi, C. Faloutsos: Recovering latent time series from their observed sums: network tomography with particle filters. *KDD* 2004: 30-39

[2] L. Bravo, W. Fan, S. Ma: Extending Dependencies with Conditions. *VLDB* 2007: 243-254.

[3] S. Chaudhuri, A. Das Sarma, V. Ganti, R. Kaushik: Leveraging aggregate constraints for deduplication. *SIGMOD* 2007: 437-448.

[4] F. Chiang, R. Miller: Discovering data quality rules. *PVLDB* 1(1): 1166-1177 (2008).

[5] C. Faloutsos, L. Li: Indexing and mining time series. Tutorial at KDD 2010: http://www.cs.cmu.edu/∼leili/TALKS/2010-KDD.

[6] S. Flesca, F. Furfaro, F. Parisi: Consistent Query Answers on Numerical Databases Under Aggregate Constraints. *DBPL* 2005: 279-294.

[7] W. Fan, F. Geerts, X. Jia, A. Kementsietsidis: Conditional functional dependencies for capturing data inconsistencies. *Trans. on Database Syst.* 33(2): 1-48 (2008).

[8] W. Fan, F. Geerts, L. Lakshmanan, M. Xiong: Discovering Conditional Functional Dependencies. *ICDE* 2009: 1231-1234.

[9] T. Fukuda, Y. Morimoto, S. Morishita, T. Tokuyama: Mining Optimized Association Rules for Numeric Attributes. *PODS* 1996: 182-191.

[10] L. Golab, T. Johnson, N. Koudas, D. Srivastava, D. Toman: Optimizing away joins on data streams. *SSPS* 2008: 48-57.

[11] L. Golab, H. Karloff, F. Korn, D. Srivastava, B. Yu: On generating near-optimal tableaux for conditional functional dependencies. *PVLDB* 1(1): 376-390 (2008).

[12] L. Golab, H. Karloff, F. Korn, A. Saha, D. Srivastava: Sequential Dependencies. *PVLDB* 2(1): 574-585 (2009).

[13] J. Kiernan, E. Terzi: Constructing comprehensive summaries of large event sequences, *KDD* 2008: 417-425.

[14] F. Korn, S. Muthukrishnan, Y. Zhu: Checks and Balances: Monitoring Data Quality Problems in Network Traffic Databases: *VLDB* 2003: 536-547.
[15] D. Neill, W.-K. Wong: Event detection: Tutorial at KDD 2009: http://www.cs.cmu.edu/∼neill/papers/eventdetection.pdf.
[16] M. Slade: Dramatic slowdown in credit card debt. New Zealand Herald, 1 May 2009. www.nzherald.co.nz.
[17] http://archive.ics.uci.edu/ml/machine-learning-databases/event-detection/CalIt2.data
[18] http://archive.ics.uci.edu/ml/machine-learning-databases/event-detection/CalIt2.events
[19] http://www.rbnz.govt.nz/statistics/monfin
[20] http://ita.ee.lbl.gov/html/traces.html
[21] gwa.ewi.tudelft.nl/pmwiki/pmwiki.php?n=Workloads.Gwa-t-1