Pattern Recognition and Anomaly Detection in Bookkeeping Data [*]

Pierre Jinghong Liang
Tepper School of Business
Carnegie Mellon University

Aluna Wang
Tepper School of Business
Carnegie Mellon University

Leman Akoglu
Heinz School and School of Computer Science
Carnegie Mellon University

Christos Faloutsos
School of Computer Science
Carnegie Mellon University

(Preliminary/Incomplete)
March 2021

Abstract

We introduce the Minimum Description Length (MDL) principle in performing tasks of pattern recognition and anomaly detection in bookkeeping data. The MDL principle underlies many machine learning applications in practice, especially in unsupervised settings. We report and summarize recently developed MDL-based computational techniques specifically designed for large volumes of transaction-level data with features such as account networks or graphs, created by double-entry bookkeeping, and the inherent account classification (assets vs. liabilities or operating vs. financing accounts). Applied to journal entry data from four different companies over an entire calendar year, these techniques are shown to be effective in recognizing patterns in the data and (thus) in spotting anomalies, as evidenced by successful case studies and recalling injected anomalies including those created by audit practitioners. It is shown that our MDL-based graph-mining solutions highlight the importance of the double-entry-bookkeeping system and an economics-based account classification. In turn, these bookkeeping features make the MDL-based and graph-mining tools valuable when working with bookkeeping data.

Keywords: Pattern Recognition, Anomaly Detection, Bookkeeping, Minimum Description Length Principle, Machine Learning, Graph mining

[*]This document and its associated presentation are prepared under specific data usage agreements (DUA). As such a portion of the paper has been withheld until a later time. Please do not distribute beyond this use. The authors gratefully acknowledge financial support and professional expertise provided by the Digital Transformation and Innovation Center at Carnegie Mellon University through its sponsored research activities. Pierre Jinghong Liang also has research affiliations with Shanghai Advanced Institute of Finance and China Academy of Financial Research at Shanghai Jiaotong University. The codes for the algorithms described in the paper are available at CompreX, CODEtect and TG-SUM. All errors are our own.

1 Introduction

Detecting anomalies among large volume of data remains a significant challenge in many fields of business practice. Within the service scope of the accounting profession, much of the advisory and assurance work would benefit from improved anomaly detection solutions. Fraud detection is one but not the only obvious example. The 2020 PwC Global Economic Crime and Fraud Survey reports that 47% of responding companies experienced a fraud in the past 24 months, six frauds on an average reported per company and a total of $42bn in fraud losses according to the report. With the 2002 passage for Sarbanes–Oxley Act, penalties for fraudulent financial activity are more severe and the oversight roles of boards of directors and the outside auditors, who review the accuracy of corporate financial statements, have also increased. According to the current auditing standard, performing financial statement audit, procedures in consideration of fraud would involve "examining journal entries and other adjustments for evidence of possible material misstatement due to fraud" (PCAOB AS 2401, paragraph 58). While not all anomalies are indeed frauds, discovering anomalies among a large volume of transaction-level data is a main first step towards detecting misstatements and frauds. As a result, professional service firms such as the Big-four accounting firms face increasing demand from both clients and regulators to improves its anomaly detection capabilities, especially with the aid of modern data mining techniques.

Published research on computer-assisted anomaly detection in large volume transaction-level data can be traced at least to Bay et al. [2006]. In this paper, we introduce the Minimum Description Length (MDL) principle in performing tasks of pattern recognition and anomaly detection in bookkeeping data. While deeply grounded in the concept of Kolmogorov complexity (Kolmogorov [1965]), a cornerstone of the theoretical foundation of information and computer science, MDL principle underlies many practical machine learning applications, especially in unsupervised settings such as anomaly detection. Since its original formulation by Rissanen [1978] in the 1970s, it has help develop many computational solutions to many practical data mining problems (see surveys by Grünwald [2007] and Faloutsos and Megalooikonomou [2007]), including recent work addressing anomaly detection problems in accounting.

Specifically, we report and summarize MDL-based computational techniques specifically designed for accounting data with features such as account networks or graphs, created by double-entry bookkeeping, and the inherent account classification. These bookkeeping features create unique challenges and opportunities to develop algorithmic solutions to real-world anomaly detection problems facing accounting professionals everyday. Three MDL-based algorithmic solutions are reported and summarized, all deployed on journal entry level data from four different companies with a total of over three hundred thousands journal entries and more than three million lines of raw data (debits and credits). Among the three algorithmic solutions, two are designed specifically to detect anomalous journal entries and one for efficient journal-entry summarization. The anomaly detection techniques are shown to be effective in spotting anomalies, as evidenced by successful case studies and recalling injected anomalies created by audit practitioners. The data summarization technique is designed to recognize patterns in the journal entries effectively to merge similarly-behaving accounts in a data-driven manner.

Aside from the technical improvement in practice, the work reported here generate insights on the design and characteristics of bookkeeping. Our MDL-based data/graph-mining solutions highlight the importance of the double-entry-bookkeeping system and an economics-based account classification. In turn, these accounting features makes the MDL-based graph-mining tools valuable when working with bookkeeping data. MDL approaches are particularly suitable for accounting data by exploiting its inherent structure and economics-based relations among attributes of accounting objects such as journal entries and classification. Specifically, double-entry bookkeeping induces a graph representation which makes the use of the modern graph-mining feasible. However, account graphs are quite unique (e.g., the existence of compound journal entries) and requires much care in the graph construction. In addition, classifications are ubiquitous in bookkeeping (current, long-term, assets, liability, operations, financing, etc.). It turns out these classifications are relevant in performing technical tasks such as pattern recognition and anomaly detection. In turn, the technical solutions developed under MDL can be used to evaluate (or quantify) alternatives classifications and aggregation schemes. In fact, in one experiment using transaction-level journal entry data, we show, quantitatively, a desirability of accounts reclassified using a scheme similar in spirit to those in Feltham and Ohlson [1995] and Nissim and Penman [2001], as compared to the existing GAAP classifications.

Combined, these specific studies point to a research approach and a few practical instances that open a new avenue to use machine learning (ML) tools, especially graph-mining, to detect anomaly better than traditional multi-variable tools. More important, the research approach allows a return to studying some practical but central tasks by the accounting practitioners, such as bookkeeping, classification and aggregation, with modern quantitative tools.

While we defer discussion of the existing work specifically related to anomaly detection to Section 3. Here, we emphasize two historically central areas of the accounting research which serve as a foundation for our current interdisciplinary work. One area is the mathematical structure of double-entry-bookkeeping, which originates in the writings of the fifteen century Italian mathematician Luca Pacioli (Pacioli et al. [1994]). Modern researchers have advanced our understanding of mathematical representation of debit-credit bookkeeping language (Ijiri [1965] and Butterworth [1972]) and its implication to modern information economics (Arya et al. [2000]). Of particular interest for our current paper is the graph structure of double-entry-bookkeeping described in Ijiri [1993] and Arya et al. [2004].

Another central area is the accounting classification which accompanies, and is necessitated by, double-entry-bookkeeping. Classification is the ubiquitous bookkeeping task in internal reporting such as the design of its Chart of Accounts (COA) as well in external reporting such as financial statement line item (FSLI) in 10-K filings. Conceptually, Ijiri [1975] pointed out that the classification can be viewed as rooted the quantification axiom (page 57), which is a foundational concept in many earlier attempts at constructing a unified conceptual framework for accounting. One challenge here is the difficulty in quantifying classification to facilitate rigorous comparisons between classification regimes; MDL approach, we believe offers one such quantification alternative.

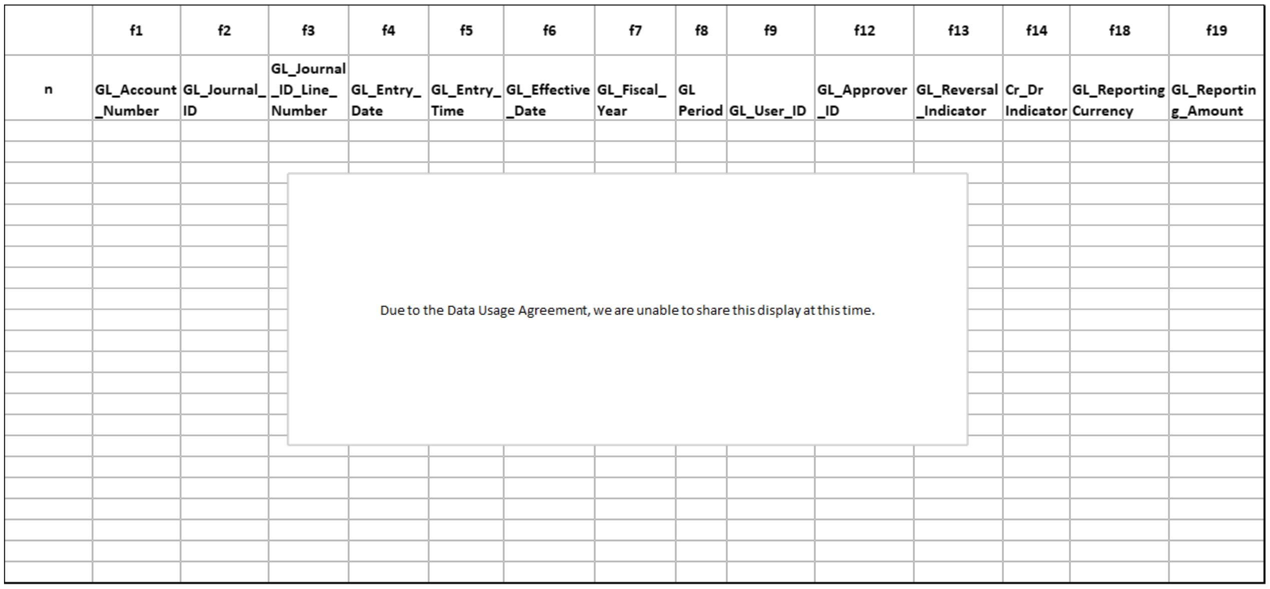
2 Motivating Use-case: mining journal entry graphs for anomalies

To begin, we describe a use-case to motivate the underlying idea of combining double-entry structure and MDL-based data mining technology to solve practical accounting problems of patterns recognition and anomaly detection in bookkeeping data. In short, we show by leveraging the essential graph data structure created by the double-entry and the inherent account-classification, the modern data mining tools based on MDL is able to achieve desirable outcomes. In this section we describe how we derived the solution.

2.1 Motivating Use Case: anomaly detection problem definition

Within an organization, the listing of each and every journal entry in the chronological order, i.e., the general journal, is the backbone of the bookkeeping system. The problem defined by the practitioners is, loosely speaking, to create a computer algorithm to traverse through all the journal entries and spot anomalous journal entries in an unsupervised manner.

A snippet of a typical data-set is shown in Figure 1. In such a data-set, each journal entry is identified by a unique journal entry ID, column "f2" called "`GL_Journal_ID`". Most data columns (or features) are **meta data**: entry data and time, effective date, user and approval pair, etc. while **relational data** consist of column "f1" specifies accounts affected by the journal entry with a companion corporate Chart of Accounts providing the name and classification of the account IDs; column "f14" specifies debit or credit; and column "f19" specifies dollar amounts of the entry (by convention in debit is coded as positive and credit as negative amounts in our data set).



	f1	f2	f3	f4	f5	f6	f7	f8	f9	f12	f13	f14	f18	f19
n	GL_Account_Number	GL_Journal_ID	GL_Journal_ID_Line_Number	GL_Entry_Date	GL_Entry_Time	GL_Effective_Date	GL_Fiscal_Year	GL Period	GL_User_ID	GL_Approver_ID	GL_Reversal_Indicator	Cr_Dr Indicator	GL_Reporting Currency	GL_Reporting_Amount

Due to the Data Usage Agreement, we are unable to share this display at this time.

Figure 1: A Snippet of A Typical Journal Entry Data-set

The following table describes the basic statistics of the four data-sets.

	number of unique accounts	number of unique data lines	number of journal IDs
Data-set 1	348	$337,508$	$92,014$
Data-set 2	$2,230$	$1,596,673$	$157,006$
Data-set 3	261	$827,448$	$40,086$
Data-set 4	142	$896,028$	$9,941$

Table 1: Data set Statistics Table

A more precise problem definition is now given.

Problem Definition 1 *For a typical journal entry data-set with both meta- and relational data component, develop a general, scalable, explainable, and unsupervised detection model. The input of the model is the journal entry data in the form of Figure 1 and the output of the model is an anomaly rank of all journal entries.*

Here are the brief explanation of each key words in the problem definition

- **Meta-data component** data about the journal entry such as entry date, time, user-approver pair, etc.
- **Relational-data component** journal entry data including accounts debited or credited and the dollar amount of each debit and credit.
- **General** the solutions will not need to be custom-made for a specific entity.
- **Scalable** the time for compute the solutions will not increase dramatically when more data are fed into the algorithm.
- **Explainable** the solution would allow users to explain why a journal entry is deemed by the algorithm as more anomalous than another.
- **Unsupervised** the solution does not need "training data" from an entity (i.e., journal entries labeled as anomalous and journal entries labeled as normal).

2.2 Motivating Use Case: CODEtect solution

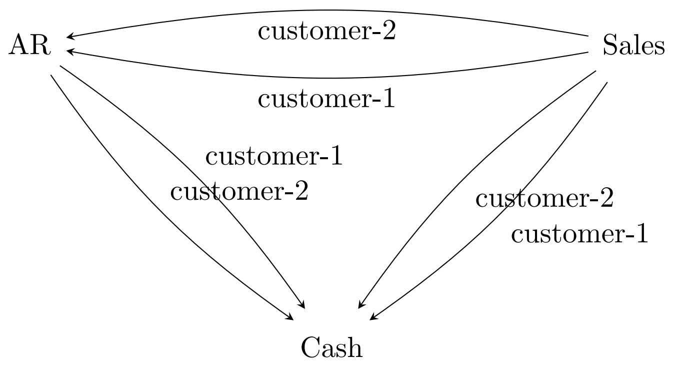
A solution to the anomaly detection problem we developed for the practitioners consists of two separate MDL-based components: one algorithm, called CompreX, is designed to handle the categorical nature of the meta-data of the journal entries while another, called CODEtect, is designed to handle the relational data. For this motivational use case illustration, we focus on the CODEtect algorithm, which require a combination of both mathematical theory of double-entry bookkeeping and modern graph mining techniques based on MDL principle.

2.2.1 Converting double-entry data into graphs

Since Renaissance Italian scholar Pacioli's 1494 work formalizing the double-entry bookkeeping practice, it has become well-known that the journal entries have a directed graph representation:

- Each distinct account is represented as a distinct node;
- A debit entry to an account is represented as an incoming edge into an node representing the debited account;
- A credit entry to an account is represented as an outgoing edge from an node representing the credited account;
- any collection of journal entries, thus, creates a network of accounts represented by a directed graph (digraph)

Here is an example of digraphs created by simple bookkeeping records based on the convention above. The three accounts, cash, accounts receivable, and sales, forms a triangle of transactions if the reporting entity aggregates both credit and cash sales into a single reported financial statement line-item (FSLI) called "Sales"



For more on the graph and matrix representation of bookkeeping data under double-entry bookkeeping, see Ijiri [1965], Butterworth [1972] Arya et al. [2000], and Fellingham [2018], among others. For the journal entries in our datasets, the relational data contains simple journal entries as well as compound journal entries where each entry contains more than two distinct accounts and with more than one debit and one credit. Graph construction is non-trivial and much work has been done and need to be done. In the real datasets, the largest single entry consists of 65 nodes and 1700 edges! Figure 2 is the graph generated by only 10 day worth of the transactions in one of our dataset.

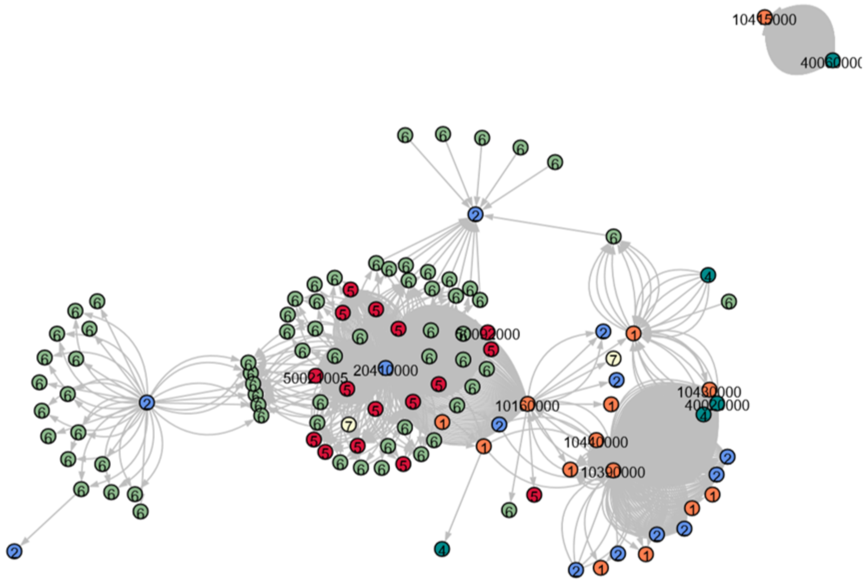


Figure 2: Bookkeeping Graph for a 10-day window of Data-set 1

When transformed, each journal entry becomes a graph: a mathematical object without losing any information contained in the relational component of the journal entry. Specifically, the journal entry database becomes a graph database as follows:

Definition 2 (Bookkeeping Graph Database) *A database $\mathcal{G}$ is a collection of J journal entries where each entry is represented as a graph $G_j = (V_j, E_j)$ with at least two nodes and one directed edge.*

- *a node $u \in V_j$ corresponds to an account such as cash or accounts receivable;*
- *a directed edge $(u, v) \in E_j$ corresponds to a credit to account u and a debit to account v;*
- *$m(u, v)$ represents the number of edges from u to v within the journal entry G_j;*
- *$\cup_j V_j$ corresponds to the set of all accounts in the company's chart of accounts (COA)*
- *$\mathcal{T}$ denote the set of account labels: $\mathcal{T} = \{assets, liabilities, equity\}$ for example;*

2.2.2 Use MDL to Mine Bookkeeping Graphs

With journal entries transformed into graphs, the anomaly detection in the relational data component of the journal entries can be restated as a graph-mining problem. From a graph-mining perspective, detecting anomalous graphs in database $\mathcal{G}$ is to discover robust patterns in the graph database that "explain" or compress the database well, and flag those graphs that do not exhibit such patterns as expected—simply put, graphs that do not compress well are more likely to be anomalous. See Akoglu et al. [2015] for a recent survey on graph-based anomaly detection.

Specifically, the graph patterns we seek are *substructures* of a graph, called motifs, which occur frequently within the input graphs. The motifs with more frequently uses in the graph dataset $\mathcal{G}$ can be used to compress the data, operationalized by encoding the existence of each such motif with a short code. Formally, the patterns we seek are modeled as a two-column Motif-table ($MT \in \mathcal{MT}$) where

- first column contains small graph structures, i.e., *motifs* (g), a connected, directed, node-labeled, simple graph, with possible self-loops on the nodes.
- second column contains the codeword $code_{MT}(g)$ (or simply $c(g)$) with length $\ell(g)$. See Figure 3 for an illustration.

The goal is then to discover a (small) set of motifs that compresses the data $\mathcal{G}$ the best. Building on the Minimum Description Length (MDL) principle Grünwald [2004], we computationally solve an optimization program to seek a motif table (MT) such that the total code length of (1) the motif table itself plus (2) the encoding of the data using the motif table is as small as possible. In other words, we are after a small model that compresses the data the most. The two-part objective of minimizing the total code length is given as follows.

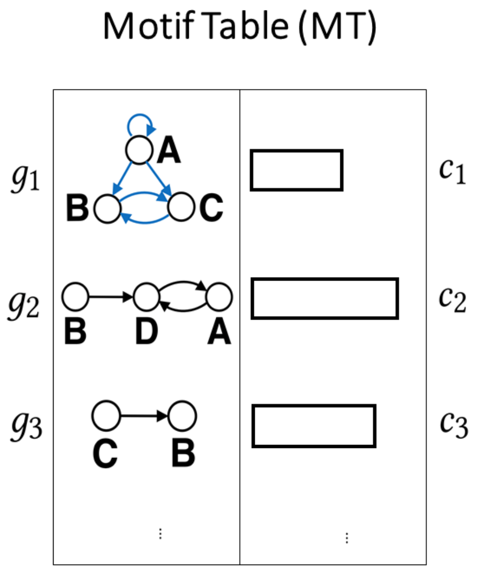


Figure 3: An MT Illustration

Definition 3 (Formal Problem Statement) *Given a set of J node-labeled, directed, multi-graphs in $\mathcal{G}$, find a motif table $MT \in \mathcal{MT}$ such that the total compression cost in bits given below is minimized:*

$$\underset{MT \subseteq \mathcal{MT}}{\textit{minimize}} \quad L(MT, \mathcal{G}) = \underbrace{L(MT)}_{\textit{model code length}} + \underbrace{L(\mathcal{G}|MT)}_{\textit{data code length}}$$

The key idea of the motif table was to economize over frequencies of sub-graphs commonly used in lots of real journal entries (leading to patterns). There are two dimensions of the optimization worth mentioning to get at the intuition of the graph mining. First, for any given MT, the efficient encoding is very intuitive: simply use Shannon coding scheme by assigning short codes to motifs or sub-graphs that occur frequently. Along the second dimension, and key to the MDL, is the tradeoff on the complexity of the model. If we choose a very long list of MT with lots of simple and also complicated motifs, the length of the data given MT may be shorter but the length to MT is longer and vice versa. MDL ensures that the combined length is minimum. So the best MT is data specific.

Use Compression to Detect Anomalies Compression based techniques are naturally suited for anomaly and rare instance detection. This is how we exploit the dictionary based compression framework for this task

- In a given motif table, the patterns with short code words with high usage represent the sub-graphs in the graph database that can effectively compress the majority of the data points. Consequently, the graphs in a graph database can be scored by their encoding cost for anomalousness.
- Formally, for every graph $G_j \in \mathcal{G}$, compute the anomaly score as:

$$score(G_j) = L(G_j|MT^*)$$

  where MT^* denote the solution to the problem Definition, the best motif table among those considered. The higher the score, the the more likely it is to arouse suspicion that it was generated by a different mechanism.

2.3 Motivational Use Case: Broad Lessons

We defer, to Section 3, reporting the performance of the CODEtect when applied to the actual journal entry datasets in our experiments, including the performance evaluation using qualitative case studies and quantitatively using embedded anomalies. CODEtect advances the growing graph-mining anomaly detection work by expanding the types of graph (e.g., those featuring multi-edges and node-labels) which can be mined. The technical write-up of CODEtect is deposited at arXiv (Nguyen et al. [2020]) and includes the source codes. Here we summarize the critical lessons learned from our exercise.

- **Double-entry bookkeeping is critical**: without the graph structure of the double-entry bookkeeping, no such advances is possible. The graph-based approach avoids the problems of feature-dimension explosion emerged when conventional unsupervised detectors are asked to discover structural anomalies such as those in the relational journal entry data.
- **Accounting classification is critical**: in generating the motif table, a critical parameter is the number of possible nodes in a motif the algorithm is asked to search. Here the classification of accounts (or nodes) plays a critical role.
- **Interdisciplinary collaboration is key** to solving practical problems facing practitioners in our profession. Three factors contributed to the initial success of the current research. First, a

valuable practical problem is presented by the industry partner who also supplied proprietary data. Deep mathematical theory of bookkeeping provided the underlying graph structure of the data and guided non-trivial graph construction tasks. Modern machine learning, especially graph-mining techniques provided fast and efficient solutions within the parameters required by the industry partner (general, scalable, and explainable unsupervised detector). With these three ingredients, we managed to combine centuries-old bookkeeping structure with modern computer science theory (MDL) and tools (graph mining) to help improve the capability of solving practical problems faced by practitioners everyday in internal and external auditing practices.

3 Background and Research Questions

In this section, we briefly provide the background on anomaly detection in accounting data in general and the theoretical idea underlying the Minimum Description Length (MDL) as an inductive inference principle in applied unsupervised machine learning.

3.1 Background of Anomaly Detection in Accounting Data

Detecting anomalous transactions is part of comprehensive external audit procedures required by standard setters. According to the Auditing Standard (AS 2401) of the PCAOB, when performing financial statement audit, procedures in consideration of fraud include "Examining journal entries and other adjustments for evidence of possible material misstatement due to fraud. ... Accordingly, the auditor should design procedures to test the appropriateness of journal entries recorded in the general ledger and other adjustments (for example, entries posted directly to financial statement drafts) made in the preparation of the financial statements" and specifically highlights computer-assisted tools to identify suspecious journal entries. For example, paragraph 61 states "in an IT environment, it may be necessary for the auditor to employ computer-assisted audit techniques (for example, report writers, software or data extraction tools, or other systems-based techniques) to identify the journal entries and other adjustments to be tested." More broadly, the document recommends that the auditors obtain an understanding of the entity's financial reporting process and the controls over journal entries and other adjustments, identify and select journal entries and other adjustments for testing, consider the characteristics of fraudulent entries or adjustments, the nature and complexity of the accounts, and journal entries or other adjustments processed outside the normal course of business.[1]

[1]Specifically, AICPA AU 316.61 states "inappropriate journal entries and other adjustments often have certain unique identifying characteristics. Such characteristics may include entries (a) made to unrelated, unusual, or seldom-used accounts, (b) made by individuals who typically do not make journal entries, (c) recorded at the end of the period or as post-closing entries that have little or no explanation or description, (d) made either before or during the preparation of the financial statements that do not have account numbers, or (e) containing round numbers or a consistent ending number." and that "inappropriate journal entries or adjustments may be applied to accounts that (a) contain transactions that are complex or unusual in nature, (b) contain significant estimates and period-end adjustments, (c) have been prone to errors in the past, (d) have not been reconciled on a timely basis or contain

Prior Computer-assisted detection work at the transaction-level include two collaborative projects involving a leading professional accounting firm and academic scholars from computer science.

Project Sherlock In Bay et al. [2006], a two-stage analytic approach is developed and applied at a disaggregated level find suspicious accounting behaviors that may be indicative of fraud or other material errors. The approach, part of the so-called Project Sherlock, first search for unusual behaviors in the data before turning them into the features of a classifier that attempts to quantify the patterns that have been found in companies with known misrepresentations. The primary challenge involves unlabeled data: for many of the training data points the class label is not known with absolute certainty. Variations of naïve Bayes approach was considered and experimented. The results indicate that explicit treatment of label uncertainty can improve performance given appropriate assumptions.

Project SNARE In McGlohon et al. [2009], while also investigating analytics that operate at a transaction level, the authors show how exploiting the link structure between accounts has the potential to greatly increase the accuracy of classification methods while making only a few assumptions. Using a graph-based approach, the SNARE (Social Network Analysis for Risk Evaluation) methodology is shown to detect related entities that may be overlooked by using individual risk scores in general ledger accounting data. However, SNARE requires externally provided data labels of suspicious transactions, making it a (semi-)supervised approach, unlike the three studies we reported here.

There are also existing work in detecting anomalies using publicly available data at the firm level. Here the goal is to detect firm-years where publicly available GAAP aggregate financial statements contains some anomaly. These work of detecting accounting irregularities using financial statements goes back at least to Beneish [1997] and Beneish [1999]. An influential subsequent work is Dechow et al. [2011] who not only developed supervised prediction model that generates a score as a signal of likelihood of misstatement but also created a comprehensive database of financial misstatements. A recent NLP-based machine-learning work along the same lines is Brown et al. [2020]. Unlike the unsupervised machine learning methods we describe in this paper, these existing work uses prior observable events such as restatement or SEC enforcement actions as labels to develop a supervised learning methodology.

3.2 Minimum Descriptions Length Principle and Applications

This section briefly describes the background of the MDL methodology used to develop unsupervised learning algorithms used in the accounting use-cases in this paper.

Originated in the work by Rissanen [1978], the Minimum Description Length (MDL) Principle is a theory of inductive inference with wide applications to statistics, machine learning and pattern

unreconciled differences, (e) contain intercompany transactions, or (f) are otherwise associated with an identified risk of material misstatement due to fraud. The auditor should recognize, however, that inappropriate journal entries and adjustments also might be made to other accounts."

recognition. Its basic idea is that the best explanation for a given set of data is provided by the shortest but lossless description of that data. In other words, a better explanation compresses the data more. Intuitively, the principle offers two insights in how to find a *Good Model* that explains the (limited) data (see [Grünwald, 2004]):

Observation 4 (MDL Insight #1) *Any **regularity** in the data can be used to **compress** the data, i.e. to describe it using fewer symbols than the number of symbols needed to describe the data literally. The more regularities there are, the more the data can be compressed.*

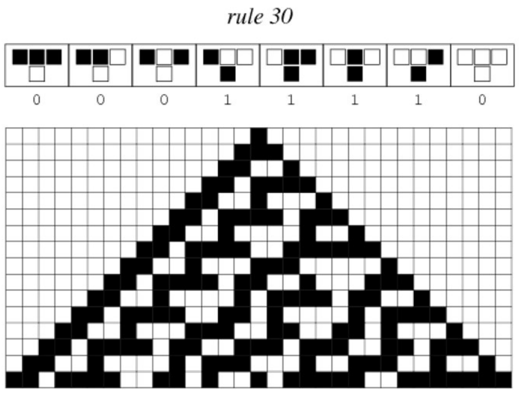
Observation 5 (MDL Insight #2) *Thus **learning** from observations becomes **finding regularities** to compress the observations; the more we are able to compress the data, the more we have learned about the data.*

Philosophically, MDL is consistent with the so-called "Ockham's Razor" named after philosopher William of Ockham (c. 1287 to 1347), who said "Nunquam ponenda est pluralitas sine necesitate": Explanations should not be multiplied beyond necessity.[2] Given as an illustrative example in Faloutsos and Megalooikonomou [2007], the pattern exhibited in the first million bits of fractional extension of beautiful number π can be compressed and expressed by a description using very few symbols (i.e., a fixed length computer program) such as the following expression to implements the Ramanujan's formula for π:

$$\frac{1}{\pi} = \frac{\sqrt{8}}{99^2} \sum_{n=0}^{\infty} \frac{(4n)!(1103 + 26390n)}{(n!)^4 396^{4n}}$$

As a second example, consider the famous Elementary Cellular Automata (ECA) rule-30, made famous by Dr. Stephen Wolfram, the creator of Mathematica. Rule-30 consists of purely deterministic instructions to generate next row from an existing row. For example, three consecutive white (or black) cells in one row generate a single white cells in the second row while two consecutive white followed by a black cell generate a single black cell, etc.. Famously,repeatedly applying rule-30 generates cell-patterns, as shown in the panel (b) in Figure 4, which appear so orderless that the properties of rule-30 rows are used to generate random numbers in Mathematica. Now let's reverse the process, supposed we are given a picture of these 250 rows without knowing where it comes from and are tasked to discover the pattern. Knowing this picture, the best pattern should be the eight rules depicted at the top of left panel of Figure 4 and the initial row (all whites with a single black cell). There maybe other ways to describe the picture but the key idea is to look for the shortest description of the picture. This is the central data-compression-based MDL idea: better models/patterns compress the data better.

[2]Cover [1999] explains that "In the end, we choose the simplest explanation that is consistent with the data observed. For example, it is easier to accept the general theory of relativity than it is to accept a correction factor of c/r^3 to the gravitational law to explain the precession of the perihelion of Mercury, since the general theory explains more with fewer assumptions than does a "patched" Newtonian theory." (page 490)



(a) First 15 Rows

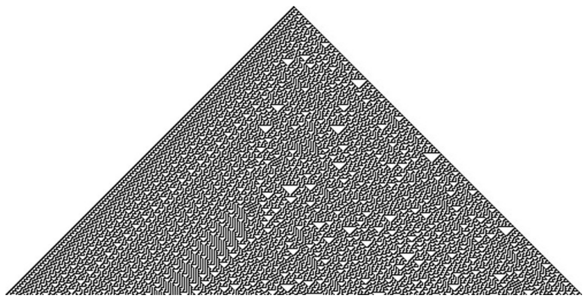

(b) First 250 Rows

Figure 4: Stephen Wolfram's ECA Rule-30

Formally, MDL is a principle used to solve a fundamental *Data Description/compression Problem* as described by Grünwald [2004]: suppose we observe a dataset (or sample) D where

$$D = (\{x_1, y_1, z_1, \dots\}, \{x_2, y_2, z_2, \dots\}, ..., \{x_n, y_n, z_n, \dots\}) \in (\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}, \dots)^n$$

We are interested in finding out a best model, $\mathcal{M}$, to explain D which may include both numeric and non-numeric (categorical) data fields.

Here the set of models to choose from is as broad as possible. The set includes parametric models such as linear, log, exponential, or k-th degree polynomials for numerical data D, as well as non-numerical lists mapping non-numerical data D into code-words, all in an effort to achieve data compression efficiently. The definition of MDL approach to solve the problem above becomes

Definition 6 (Two-part MDL Grünwald [2004]) *Let $\mathcal{M}^{(1)}, \mathcal{M}^{(2)}, ...$ be a list of candidate models (e.g., $\mathcal{M}^{(1)}$ may be a set of polynomial models while $\mathcal{M}^{(2)}$ may be a set of power function models, $\mathcal{M}^{(3)}$ may be a set of possible multi-column lists), each containing a set of point models (e.g., a particular polynomial model). The MDL point model $\mathcal{M} \in \mathcal{M}^{(1)} \bigcup \mathcal{M}^{(2)} \bigcup ...$ to explain the data D is the one that minimizes*

$$\sum L(M) + L(D|M)$$

where $L(M)$ is the length, in bits, of the description of the model; and $L(D|M)$ is the length, in bits, of the lossless description of data given the point model.

MDL has its intellectual roots in the concepts of Kolmogorov Complexity due to Solomonoff [1964], Kolmogorov [1965], and Chaitin [1969]. In Appendix A, we describe the connection between the Kolmogorov Complexity and MDL methodology. Faloutsos and Megalooikonomou [2007] surveyed much of the machine learning work and argued that "data mining, in its general case, is equivalent to compression and Kolmogorov complexity" (page 4). These work includes supervised learning such as classification, regression and unsupervised learning pattern discovery, clustering, forecasting and outlier detection. The desirable properties of MDL include avoiding overfitting automatically, free of a prior distribution, and good predictive performance (see Grünwald [2007]).

Operationally, to apply the MDL principle to a learning task (such as pattern mining), one might proceed in three main steps, as summarized by Galbrun [2020]

- **Step 1: pattern language** This is the M part: a model M is simply a language of describing the data pattern. Critically, the language constitutes a structure of interest given the data such as regular patterns. Examples are some functional form describing precise relation among variables. A more general, and commonly used, example is creating a specific dictionary patterns useful in describing the data such as complex objects such as texts or graphs. Typically, such a dictionary provides the mapping between patterns and codewords, typically referred to as the *Code Table*. The MDL approach seek to find, in part, for the best such code-tables from among possible such tables;[3]

- **Step 2: data encoding scheme** This is the $L(\dots)$ part. A suitable encoding scheme to encode the data using code-table is designed, with prefix-free codes as basic ingredients.[4] Given a code-table, this is fairly straightforward: simply replacing occurrences of the patterns in the data by their associated codewords. Count the times each pattern x_i is used in the description of the data D with model M, denoted as $usage_{D,M}(x_i)$, the Shannon-Fano entropy-style coding assignment is:

$$L(x_i) = -log_2 \frac{usage_{D,M}(x_i)}{\sum_{x_j \in M} usage_{D,M}(x_j)}$$

The requirement that the encoding be lossless means that elementary patterns must be included in the model, to ensure complete coverage.

- **Step 3: search and optimize** design a search algorithm to identify in the data a collection of patterns that yields a good compression under the chosen encoding scheme. The MDL principle provides a basis for designing a score for patterns, but no way to actually find the best patterns with respect to that score. The space of candidate patterns, and even more so the space of candidate pattern sets, is typically extremely large, if not infinite, and rarely possesses any useful structure. Hence, exhaustive search is generally infeasible, heuristic search algorithms are employed, and one must be satisfied with finding a good set of patterns rather than a globally optimal one.

[3]In cases where parametric functions are used as models, the code-table becomes a pre-existing algorithm computer uses to generate the corresponding functional outputs for a given set of parameters such as the degrees of the polynomial. Here even for functions with the same number of parameters, the length of the model can be difference. In a well-known example, Myung et al. [2000] shows that while a power-model $y = ax^b + Z$ where $Z \sim N(0, \sigma^2)$ produces a better fit for the data actually generated by a log-model $y = aln(x + b) + Z$ where $Z \sim N(0, \sigma^2)$ than the fit produced by the log-model (the true data generating process!). However, it is shown that the power model is more flexible, complex and costs longer bits to encode; so an MDL approach would favor the simpler log-model based on the data.

[4]Prefix-free codes is a type of code system requires that there is no whole code word in the system that is a prefix (initial segment) of any other code word in the system, see Cover [1999] for more details.

In the end, Faloutsos and Megalooikonomou [2007] conclude that "several core aspects of Data Mining, like classification, clustering, forecasting, outlier detection, are all essentially related to compression. ... The compression view point (MDL etc) leads to elegant, parameter-free solutions to clustering, graph partitioning, distance-function design" (page 18). Figure 5 illustrates the basic idea of inferring model from data using the MDL methodology.

To summarize, the basic ideas behind MDL are that the main task of "learning" is to separate structure from noise; regularity from anomaly; meaningful information from accidental information; and at the technical level, the compressable versus the uncompressable portion in the observed data. This data-compression idea is not dissimilar to economist and future Nobel laureate Christopher Sims' idea about economic theory and methodology; Sims [1996] remarked "[a] good theory must not only display order in the data (which is the same thing as compressing it), it must do so in a way that is convincing and understandable to the target audience for the theory" (page 106) and Ockham's-razor-like principle of "a simple theory is preferable to a complicated one if both accord equally well with the data, making the simpler one a more thorough data compression." (page 110)

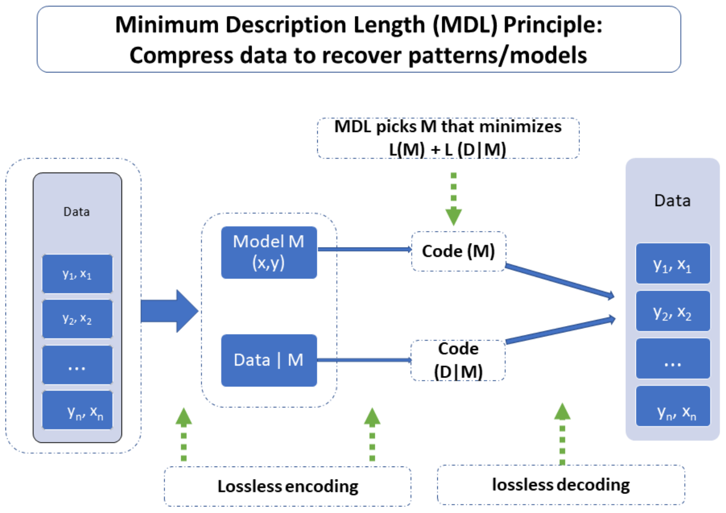


Figure 5: Illustrative Diagram of MDL

At a deeper level, the MDL philosophy recognizes that models as nothing more than languages (codes with a dictionary) for describing useful properties of the data. "Noise" is defined relative to the model as the residual number of bits needed to encode the data once the model is given. Thus, noise is *not* a random variable in an absolute sense: it is a function only of the chosen model and the actually observed data. Ultimately MDL embodies an inductive inference exercise indeed.

3.3 Using MDL for Pattern Recognition and Anomaly Detection in Bookkeeping Data

In this paper, we show that MDL provides effective solutions to detect anomalies in bookkeeping data, leveraging both categorical and graph features of the data. Specifically, in our unsupervised use-cases, we are interested in exploring the following two broad and related questions.

- Do MDL-based detection approaches improve pattern recognition and anomaly detection in bookkeeping data?

- Which and how bookkeeping features are important in pattern recognition and anomaly detection in bookkeeping data?

To address these questions, we deploy MDL-based detection tools to real-world transaction-level bookkeeping data to evaluate if and how these tools are effective in improving our ability to detect irregularities in accounting data. Formally, consider a bookkeeping dataset as a database D is a collection of n *journal entries* where each entry has m column features (such as f_1 is effective date; f_2 is name of the approver; f_3 account debited;...). Each feature $f \in \mathcal{F}$ has a domain $dom(f)$ of possible values; for example, there may be 400 distinct accounts and 10 distinct approvers, etc. in which case we says the $arity$(accounts debited) $= 400$.

Given such a bookkeeping database, the tasks of patterns recognition and anomaly detection using MDL tools involves the following steps:

Figure 6: The Code Table Illustration

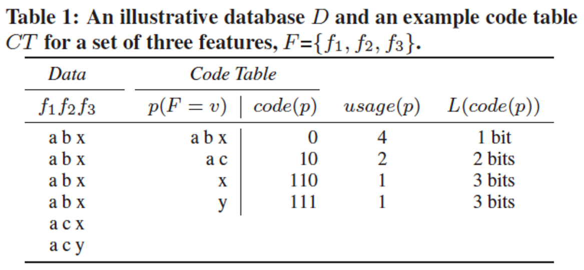


Table 1: An illustrative database D and an example code table CT for a set of three features, F=$\{f_1, f_2, f_3\}$.

Data	*Code Table*			
$f_1 f_2 f_3$	$p(F = v)$	$code(p)$	$usage(p)$	$L(code(p))$
a b x	a b x	0	4	1 bit
a b x	a c	10	2	2 bits
a b x	x	110	1	3 bits
a b x	y	111	1	3 bits
a c x				
a c y				

Step 1: pattern language Creating a dictionary listing useful patterns and assigning code-word length to each pattern listed. Denote the code-table as CT. Since optimal table is not known initially, researchers need to specify the universe of all possible patterns, the basis for all possible code tables. For example, in the transaction meta data, the language to describe the journal entry patterns can be formalized as a set of items where an item is a feature-value pair such as $\{\text{account-debited} = \text{cash}\}$. Using items so-defined, a pattern (or itemset), denoted $p \in \mathcal{P}$, can be the following combination of a few items: $\{\text{account-debited} = \text{cash}; \text{approver} = \text{doe}; \text{effective date} = \text{weekday...}\}$. The dictionary (or code-table) in our bookkeeping setting is a two-column list of optimally chosen patterns (i.e., a list of p's) matched with codewords (i.e., $c(p)$'s) optimally assigned to the patterns. See for Figure 6 an simple illustration.

Step 2: data encoding scheme Design an encoding scheme to encode the given transaction data using a given code-table efficiently. A computer program would traverse through all transaction data and convert each record into their corresponding code-word.

Step 3: search and optimize Design a search algorithm, to find the optimal code-table CT^* such that the objective $L(CT) + L(D|CT)$ is minimized. Intuitively, the more frequent a pattern repeats in the data, the more likely it should be listed in the optimal table and the shorter code-word is assigned to it. The tradeoff in the optimization exercise is that a complicated model or code-table (which lists too many potential patterns including those with infrequent occurrences) has the advantage of a short data description given model, $L(D|CT)$, but suffers from a longer $L(CT)$ when the model itself is encoded. The optimal code-table will have the property of minimum description length indeed.

Step 4: compute anomaly scores After finding the best code-table, each transaction is converted into their corresponding code-word using the *optimal code-table.* Since these code-words can be measured by their length (e.g., in bits), the code-table generates a rank-list of each transaction according to the code-length. The higher the code-length, the more likely the transaction is irregular. Formally, let CT^* denote the optimal code-table, for every journal entry $t \in D$, compute the anomaly score:

$$score(t) = L(t|CT^*) = \sum_{p \in cover(t)} L(code(p)|CT^*)$$

where $cover(t)$ is the unique set of patterns $\{p_1, p_2, ...p_{k_t} | p_i \in \mathcal{P}, i = 1, 2, \ldots, k_t\}$ that completely describe the journal entry t. The higher the score, the the more likely that it was generated by a different mechanism.

In the next section, we describe how the MDL tools are deployed to mine bookkeeping patterns and spot abnormal journal entries, tested on four sets of transaction-level level journal entry data from four real-world companies. Of special interest is how data features inherent to accounting (such as double-entry-bookkeeping) informs how MDL-based approaches can be deployed successfully.

4 Bookkeeping Data and Computational Solutions

4.1 Bookkeeping Data

Our industry partner provides us the entire set of the general ledger (GL) journal entries for a single year for four different companies. We split each dataset into two parts: meta data and graph data. Briefly meta data of the journal entries capture meta information about each entry and appear in a tabular format with each entry is recorded as a row of both numeric and categorical values. The graph data captures the relational information (connecting accounts) in each entry and appear in a graph format where each entry is recorded as a small graph with nodes (accounts) and edges between nodes (the debit/credit flows of the entry). Table 4 provides detailed statistics of the four data sets.

Meta data The meta data components of the each journal entry dataset vary slightly but typically include, for each journal entry, journal entry description, entry date, effective date, user ID, approver ID, reversal indicator, and local currency. In addition, our feature engineering generated additional features including number of distinct accounts in a journal entry, weekday, entry-date-to-effective-date difference, total entry amounts, and indicators of assets, liabilities, revenue, expenses, or equity, etc..

Relational data The relational data components of the journal entry data include accounts debited or credited and the dollar amount of each debit and credit. As described in the motivational use case (in section 2), these data are converted into graphs to leverage the graph mining technology.

Now we describe the following three instances of MDL-based tools applied to the our bookkeeping data

1. **CompreX**: Categorical-data-mining tool to detect anomalies in journal entry meta-data

	Dataset-1	Dataset-2	Dataset-3	Dataset-4
# of Features (columns)	17	18	19	21
# of Journal Entries	$92,014$	$157,006$	$40,086$	$9,941$
# of Entry lines	$337,508$	$1,596,673$	$827,448$	$896,028$
# of unique GL accounts	348	$2,230$	261	142
Fiscal date Range	Jan-1st to Dec-31st	Jan-1st to Dec-31st	Jan-1st to Dec-31st	Jan-1st to to Jun-30th
# of unique user/approver	16/NA	27/NA	17/11	62/45
Max/Total debits (=credits)	\$90M/3.8B	\$64M/8.2B	\$25M/343.5M	(FX)1B/345B

Table 2: Data set Statistics Table

2. **CODEtect**: Graph-mining tools to detect anomalies in account graphs based on journal entry relational-data (described in the motivational use case in section 2)
3. **TG-sum**: Graph-mining tools to discover efficient account classification regimes in account graphs based on journal entry relational-data

For each, we illustrate how MDL is used that leads to general, scalable, interpretable tools to learning tasks. All three properties are important for practical use as well as informing continued research.

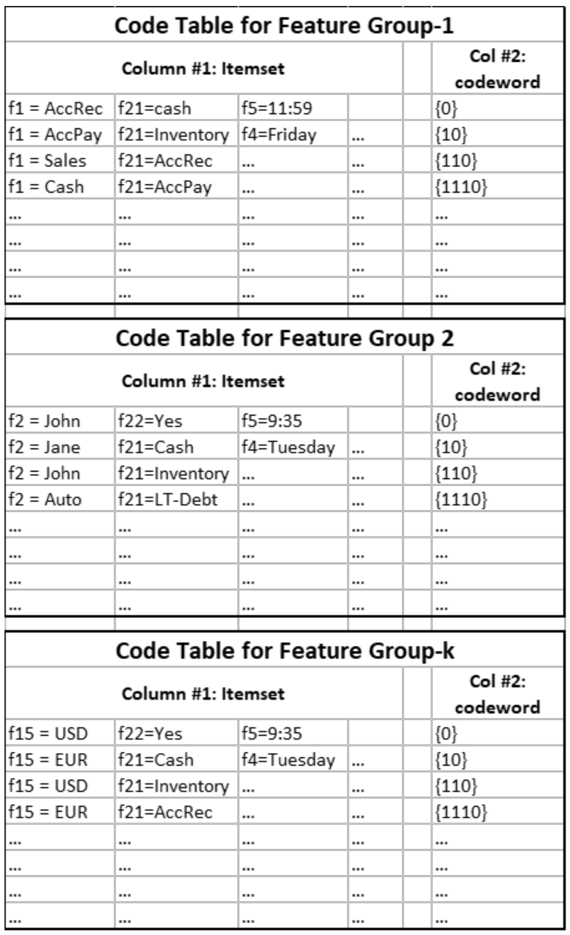
- these tools are *general*, meaning no specific calibration to specific company or industry is needed;
- these tools are *scalable*, meaning large dataset does not require computing resources at an increasing rate; and
- these tools are *interpretable*, meaning the patterns discovered and the anomalies spotted can be interpreted using the code-table and its implied frequencies .

4.2 Use MDL approach to Detect Meta-data Anomalies (CompreX experiments)

In this MDL application, we seek to discover patterns in the meta data portion of the journal entries. The challenge we face is the dimension explosion given the (1) the number of the meta-data categories, (2) size of distinct values in each categories; and (3) the number of transactions in the dataset. Additional challenges comes from added features generated by informative feature engineering by combining existing features such as weekday, weekend values for the entry/effective date variables in the raw datasets.

To proceed, we first develop the language to describe the meta-data patterns by the way of a Code Table (CT). Our approach exploits correlation among existing features by building multiple, probably smaller, code tables for each highly correlated group of features instead of a single table for all features. The code table for journal entry data looks like Figure 7.

Figure 7: Illustration: CompreX Code-tables



Code Table for Feature Group-1

Column #1: Itemset					Col #2: codeword
f1 = AccRec	f21=cash	f5=11:59			{0}
f1 = AccPay	f21=Inventory	f4=Friday	...		{10}
f1 = Sales	f21=AccRec	...	...		{110}
f1 = Cash	f21=AccPay	...	...		{1110}
...	...	...	...		...
...	...	...	...		...
...	...	...	...		...
...	...	...	...		...

Code Table for Feature Group 2

Column #1: Itemset					Col #2: codeword
f2 = John	f22=Yes	f5=9:35			{0}
f2 = Jane	f21=Cash	f4=Tuesday	...		{10}
f2 = John	f21=Inventory	...	...		{110}
f2 = Auto	f21=LT-Debt	...	...		{1110}
...	...	...	...		...
...	...	...	...		...
...	...	...	...		...
...	...	...	...		...

Code Table for Feature Group-k

Column #1: Itemset					Col #2: codeword
f15 = USD	f22=Yes	f5=9:35			{0}
f15 = EUR	f21=Cash	f4=Tuesday	...		{10}
f15 = USD	f21=Inventory	...	...		{110}
f15 = EUR	f21=AccRec	...	...		{1110}
...	...	...	...		...
...	...	...	...		...
...	...	...	...		...
...	...	...	...		...

Once the data features are given and the general two-layer design of the code tables is set, the CompreX algorithm will automatically discover the best set of smaller code-tables and optimize the code assignment within each table based the actual dataset, following the steps laid out in subsection 3.3 all the way to assigning anomaly scores. Technical details of the CompreX algorithm can be found in the appendix and in Akoglu et al. [2012].

4.3 Use MDL approach to Detect Graph Anomalies (CODEtect Experiments)

This use-case has been discussed in section 2 as the motivational example. Briefly, in this MDL application, we seek to discover patterns in the graph data extracted from the journal entries. The challenges we face is the complexity of describing the graphs. In our case, each graph extracted from a journal entry contains not only typical graph features such as (1) the number of distinct nodes (accounts) and (2) specific edges (debits and credits) between nodes (accounts). Each transaction graph also contains additional graph features such as (3) node labels (classification of GL accounts as Balance-sheet or Profit-&-Loss; or assets or liability; or even further individual FSLIs); (4) directed edges represented debit and credit legs of the entry; and (5) multiplicity representing multiple debits and credits between the same pair of accounts within the same journal entry.

4.4 Use MDL Approach to Evaluation Account Classification (TG-sum Experiments)

From the practitioners perspective, a natural next question after anomaly detection is explaining and acting on the detected anomalies. Here, classification becomes an important issue. At the GL account level, accounts classification is achieved by the corporate chart of accounts (COA). A business entity's Chart of Accounts lists, and also pre-assigns a label to, each distinct account used in its ledgers. Such labeling helps companies prepare their aggregate GAAP financial statements (FS). To comply with the GAAP, individual accounts in the corporate COA are classified into summary categories such as assets, liabilities, and equity. In fact, GAAP structure of accounts is a hierarchy of classifications: assets are further classified into current, long-term assets, etc. Drilling down to the within-firm level, distinct accounts listed in an organization's (COA) are classified into a single financial statement line-item (FSLI). For example, every single bank account are labeled as "Cash and Cash Equivalents", a typical FSLI. Frequently, many accounts are grouped into "Other assets" or "Other liabilities" as FSLI. In the US, FS captions are not uniform across corporations.

Given a chart of accounts already in place, the problems faced by practitioners are to evaluate, in a data-driven way,

1. whether there is a classification error in a journal entry detected as an anomaly. That is, is the suspected error in the journal entry possibly an account from the wrong class (current asset, long-term liability) is debited or credited?
2. are there a pair or a small set of accounts more similar with each other as they are being used as expected by the COA design?
3. are there a pair or a small set of accounts similar with each other which is unexpected by the COA design?
4. how well does the current COA organize the transactions as they are used in the data?
5. finally, are there alternative classifications which would capture the transaction better in a data-driven way?

The last two questions are important from an information design perspective. As organizations evolve through introducing new products and merger/acquisition, an open question is to quantitatively evaluate the effectiveness and efficiency of an existing classification regime (i.e., a COA table). This is especially important when organizations change making COA consistency across units desirable and when firms face regulatory environment changes regarding reporting. A data/usage based evaluation approach can be useful in finding solutions to (1) evaluate the efficiency of the existing Chart of Account (COA) classification and (2) suggest a data-driven reclassification of COA based on actual uses. If successful, the tools can be applied to improve audit planning by evaluating clients' existing account classifications, and to help improve decision making based on better classified COA.

We translate these ideas into a graph mining problem as follows

- First, we construct a single large graph, $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, by combining all the J journal entries in the Bookkeeping Graph Database in Definition 2. Compute and denote $L(\mathcal{G})$ as the total description length of the uncompressed graph $\mathcal{G}$;

- Define a *Summary graph* $\mathcal{G}_s = \{\mathcal{V}_s, \mathcal{E}_s\}$ where each node $v_s \in \mathcal{V}_s$, which we call a super-node, in summary graph $\mathcal{G}_s$ is constructed by merging a distinct set of nodes ($v \in \mathcal{V}$) in the original graph $\mathcal{G}$. Only a finite super-node types, called the *glyph* of node v_s, are feasible such as clique, stars, disconnected, ete.. Denote $L(\mathcal{G}_s)$ as the total description length of the summary graph $\mathcal{G}_s$; See Figure 8 for an illustration of the original and summary graphs.

- Define unique and unambiguous decompression of summary graph $\mathcal{G}_s$ as $\mathcal{G}'(\mathcal{G}_s) \equiv dec(\mathcal{G}_s)\{\mathcal{V}, \mathcal{E}'\}$ where the decomposition graph matches the original graph exactly in the set of nodes ($\mathcal{V}$) but not necessarily the edge set (i.e., $\mathcal{E} \neq \mathcal{E}'$). Compute and denote $L(\mathcal{G}|\mathcal{G}'(\mathcal{G}_s))$ as the total description length of the original graph $\mathcal{G}$ given the summary graph $\mathcal{G}_s$ Intuitively, $L(\mathcal{G}|\mathcal{G}'(\mathcal{G}_s))$ is needed to recover the original graph $\mathcal{G}$ from the decompression of the summary graph $\mathcal{G}_s$;

- For a given large transaction dataset $\mathcal{G}$, compute the compression ratio of a summary graph $\mathcal{G}_s$ as
$$\text{Compression ratio} = \frac{L(\mathcal{G}) - (L(\mathcal{G}_s) + L(\mathcal{G}|\mathcal{G}'(\mathcal{G}_s)))}{L(\mathcal{G})}$$

- See appendix for formal problem statement and solution approach and a technical paper Berberidis et al. [2020] is under blind review submission;

Ideally, accounts of the same classification (or label) should behave similarly in the system than accounts with different lables. This behavior can be discerned from the real-world usage data, such as the transactions graph, where accounts are connected through credit/debit relations. Under a more suitable labeling, the accounts with the same label should have more structural similarity and yield better compression.

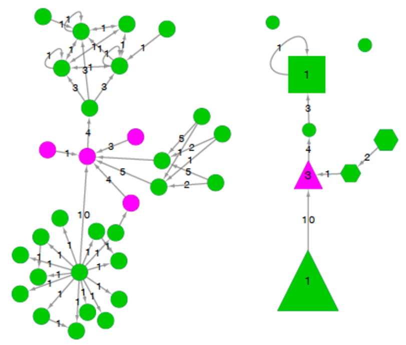


Figure 8: Summary Graph Illustration:
Input (left) and summary/super-graph (right).
E.g., triangle super-node represent star-shape subgraphs; squared-shaped represent cliques, size of the supper-nodes indicate number of original nodes in a super-node

In this application, we use TG-sum approach for the applied task of evaluating a preexisting node labeling, i.e., the set of types pre-assigned to the nodes in a graph. That is, instead of searching for the best summary graph (which is solved in the technical paper Berberidis et al. [2020]), we use the computational approach to evaluate the efficiency of the existing node (account) classification such as GAAP classification. It is important that GAAP classification is never used in the design and implementation of TG-sum, which now can serve as an independent evaluator of the GAAP or other classifications. Specifically, we consider two classifications

- **Financial Statement Classification (GAAP)** organizes the individual accounts typical GAAP categories such as inventory, prepaid expense, accounts payable, etc. For each dataset, we use specific classification given the COA given with the data.
- **Economic Bookkeeping Classification (EB)** organizes the individual accounts into operating versus financing and long versus short term accounts, all together fourteen categories.[5]

The idea is to compute the compression ratio under each of the classification regimes for each dataset and compare the compression ratio between the two regimes for the same dataset. A higher compression rate would indicate that the classification is aggregating the information in the data in a more efficient manner. All in all, TG-sum can be employed as a data-driven way to quantify the connection between a labeling and the actual role of nodes in practice as reflected in the data.

5 Results and Evaluations

Due to the restrictions under the Data Usage Agreement, at this time, we are unable to share the detailed results for some experiments we run. They will be shared if and when our displays of results are approved by our industry partner. Here we describe the several steps we evaluate the performance of our solutions.

5.1 Detecting Anomaly using Meta-data and Graph Data:

A critical challenge in any un-supervised anomaly detection is the lack of labels (or ground truth). In the bookkeeping data we work with, this means we do not have the true anomaly-status (yes/no) information for each journal entry. This is understandable in that the industry partners would not have looked at every transaction and determine if each is an anomaly or not. For those journal entries sampled and investigated during the substantive testing during the audit process, labels would have been generated and known to the engagement team. However, professional conduct and regulation imply that these labels are not to be disclosed to researchers like us.

Given the constraint on true labels, we take three approaches to evaluate the performance of our detection solutions.

[5]This classification is inspired by the reformulation of GAAP financial statement work by Professor Stephen Penman of Columbia University (Penman and Penman [2007]). Specifically, these fourteen categroies are cash, short-term operating assets, short-term financial assets,long-term operating assets, long-term financial assets, short-term operating liabilities, short-term financial liabilities,long-term operating liabilities, long-term financial liabilities, operating revenue, operating expense, non-operating gains and losses, contributed equity, and other non-earned equity.

Quantitative: injected anomalies designed by experimenter We, the experimenters, first create artificial errors in, or other alterations to, the original data, especially its graph features. We then measuring quantitatively how the CODEtect algorithms perform in terms of precision at top-k for $k = \{10, 100, 1000\}$, AUC, and average precision (AP). Two types of graph anomaly injections are performed: (a) path injections where a random edge of a journal entry is deleted and two or three path are added connecting the two nodes of the deleted edge; and (b) type injections where a random node in a journal entry is replace with a node of a different type. Performance is compared against existing anomaly detectors such as Isolation Forest (Liu et al. [2012a]) and SUBDUE (Noble and Cook [2003]), among others. Note that this quantitative evaluation is a conservative evaluation of the algorithm performance as it ignores any true anomalies in the original data.

Quantitative: injected anomalies designed by practitioners Our industry partners requested their own audit professionals to create a very small number of journal entry anomalies to be injected into the one of our four dataset. We them measure recall performance of our algorithm similar to the previous quantitative exercise. Again, this quantitative evaluation is also a conservative evaluation of the algorithm performance as it ignores any true anomalies in the original data.

Qualitative: detected anomaly case studies Here we studies the top 20 cases that CODEtect spots for one dataset and analyze the economic and bookkeeping meaning of each journal entry. Since the identity of the company is unknown to us and quite a few fields in the data have been anonymized, these case studies are not exact replica of a journal entry test in a true audit. Nevertheless, these case studies can spot obvious violation of accounting principles and odd features requiring additional auditor attention.

The incremental value of Double-entry-bookkeeping While we are not able to, at this time, share detailed results of our experiment, we can report that the initial results from these MDL-based un-supervised pattern recognition and anomaly detection approaches are very encouraging based on both quantitative and qualitative performance evaluation and benchmarking against existing approaches, as well as feedback from our industry partner. For accounting researchers, we emphasize an encouraging insight due to the inherent graph structure of the double-entry bookkeeping baked into the accounting data.

Specifically, in our evaluation using industry-partner-injected anomalies, we notice that CODEtect identifies different injected anomalies than does CompreX. This is good news generally but highlights how the graph structure, which is only picked up by CODEtect, is an important source of knowledge to be learned and mined. This is the same graph structure underlying data created only through the double-entry bookkeeping. These initial results points to the desirability of the need to study the graph structure of accounting data. The benefit not only stems from accessing the powerful graph-mining techniques in modern computer science but also from uncovering hidden properties of bookkeeping data not easily modeled in non-graph format.

5.2 TG-Sum: Pattern Recognition of Account Networks

To compare GAAP and EB classifications, we run TG-sum on each dataset using each labeling separately, and record the compression rate

- record the compression rate ("actual") and shuffle the labels randomly, and run TG-sum again and record teh compression rate ("shuffled")
- Actual values are not directly comparable. What is comparable is the difference from Shuffled: how much the labeling can improve on top of the random assignment of the same set of labels. Further, even this absolute difference is not fair to compare: it is harder to compress a graph that has been compressed quite a bit even further. So we constructed a Normalized gain:

$$\text{Normalized Gain} = \frac{Acutal - Shuffled}{1 - Shuffled}$$

- As the following results show, for EB, Shuffled rates are already high. Improving over Shuffled even by the same amount proves EB superior to GAAP from a compression perspective.

Dataset	Labeling of	Shuffled	Actual	normalized Gain (%)
Data-set 1	EB	.28	.35	5.6%
Data-set 1	GAAP	.25	.27	2.7%
Data-set 2	EB	.36	.47	17.0%
Data-set 2	GAAP	.16	.27	13.0%
Data-set 3	EB	.33	.42	13.7%
Data-set 3	GAAP	.31	.39	12.0%

Table 3: Evaluating Compression Rate of EB versus GAAP Account Classifications

5.3 Benchmark Comparisons

As is standard in algorithmic work, benchmark comparisons are conducted where the same data is subjected to investigation by alternate detection algorithms. Here we list competing algorithms used in each of the two approaches we developed specifically for the bookkeeping data. The three technical papers provides additional details for these benchmarks.

One note on TG-sum is its versatility. TG-sum is the first known graph summarization method that can simultaneously handle node-Types, edge Directions, edge Multiplicities, and Self-loops if any, as well as any combination thereof. In fact, these features are precisely what define bookkeeping graph. As a benchmark, alternative graph summarization approaches lack the ability to handle at least one of the features listed above.

CODEtect	TG-sum
SMT	Navlakha et al. [2008]
SUBDUE	SUBDUE
iForest	SNAP
iForest+G2V	CoSum
iForest+DGK	VoG
ENTROPY	GraSS
MULTI-EDGES	Liu et al. [2012b]

Table 4: Benchmark Algorithms Used

6 Conclusions

We introduce the Minimum Description Length (MDL) principle in performing tasks of pattern recognition and anomaly detection in bookkeeping data. The MDL principle underlies many machine learning applications in practice, especially in unsupervised settings. We report and summarize recently developed MDL-based computational techniques specifically designed for large volumes of transaction-level data with features such as account networks or graphs, created by double-entry bookkeeping, and the inherent account classification (assets vs. liabilities or operating vs. financing accounts). Applied to journal entry data from four different companies over an entire calendar year, these techniques are shown to be effective in recognizing patterns in the data and (thus) in spotting anomalies, as evidenced by successful case studies and recalling injected anomalies including those created by audit practitioners. It is shown that our MDL-based graph-mining solutions highlight the importance of the double-entry-bookkeeping system and an economics-based account classification. In turn, these bookkeeping features make the MDL-based and graph-mining tools valuable when working with bookkeeping data.

Appendix A. Kolmogorov Complexity and MDL

We follow Faloutsos and Megalooikonomou [2007] in describing the basic definitions and theorems as the theoretical foundations for compression based machine learning tools such as minimum description length (MDL) principle.

Definition 7 (Conditional Kolmogorov Complexity) *Let p be a computer program, $\mathcal{U}$ be a universal Turing machine, and $\mathcal{U}(p)$ denote of output of $\mathcal{U}$ when presented with p. The conditional*

Kolmogorov complexity $K(x|y)$ of a string $x \in \mathcal{X}$ under condition $y \in \mathcal{Y}$ is

$$K_{\mathcal{U}}(x|y) = \min_{p: \mathcal{U}(p,y)=x} \ell(p)$$

the shortest description length of x over all descriptions interpreted by computer $\mathcal{U}$.

Definition 8 (Unconditional Kolmogorov Complexity) *The unconditional Kolmogorov complexity of a string x is defined as*

$$K_{\mathcal{U}}(x|y) \quad \textit{where} \quad y = \emptyset$$

- Definition due to Solomonoff (1960, 1964), Kolmogorov (1965), and Chaitin (1966)
- *Universality of $K(x)$*: for any other computer $\mathcal{A}$, $K_{\mathcal{U}}(x) \leq K_{\mathcal{A}}(x) + c_{\mathcal{A}}$ where $c_{\mathcal{A}}$ is a constant

Intuition : The definition of Kolmogorov complexity agrees with our intuition:

- Using an example given in Faloutsos and Megalooikonomou [2007]: the first million bits of fractional extension of π can be implemented by Ramanujan's formula

  $$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{0}^{\infty} \frac{(4n)!(1103 + 26390n)}{(n!)^4 396^n}$$

  which is probably the shortest and fastest converging formula for π [Schroeder, 2009].

Intuition In Shannon's theory, descriptive complexity of such an object depends on the probability distribution. But Kolmogorov went further:

- defined the algorithmic (descriptive) complexity of an object to be the length of the shortest binary computer program that describes the object;
- Kolmogorov complexity of an object dispenses with the probability distribution;
- Kolmogorov made the crucial observation that the definition of complexity is essentially computer independent (*universality of K(x)*);
- Thus, the shortest computer description acts as a universal code which is uniformly good for all probability distributions (Solomonoff's original idea);
- In this sense, algorithmic complexity is a conceptual precursor to entropy.

Kolmogorov Complexity and Entropy

- Remarkably, the expected value of the Kolmogorov complexity of a random sequence is close to the Shannon entropy

Theorem 9 (Kolmogorov Complexity and Entropy) *Let $\{X_i\}$ be drawn i.i.d according to the probability mass function $f(x)$, $x \in \mathcal{X}$ where $\mathcal{X}$ is finite alphabet. Let $f(x^n) = \Pi_{i=1}^{n} f(x_i)$. Then there exists a constant c such that*

$$H(X) \leq \frac{1}{n} \sum_{x^n} f(x^n) K(X^n|n) \leq H(X) + \frac{(|\mathcal{X}|+1)logn}{n} + \frac{c}{n}$$

and consequently

$$E\left[\frac{1}{n} K(X^n|n)\right] \to H(X)$$

- It is an amazing fact that the expected length of the shortest binary computer description of a random variable is approximately equal to its entropy.

Noncomputability Theorem In the well-known *Halting problem* in theoretical computer science, loosely stated, for any computational model, there is no *general algorithm* to decide whether a program will halt or not (go on forever). This claim was proved by Alan Turing in 1936. Conceptually, this problem is the computer science's analog to math's *Incompleteness theorem* proved by Gödel in 1931. As a result, it has a negative implication about the computability of Kolmogorov complexity

Theorem 10 (noncomputability of Kolmogorov complexity) *The Kolmogorov complexity of an arbitrary string $x, K(x)$ is not computable.*

To determine K(x) we must execute every program and collect all that stop and compute x, choosing the one that has the smallest size. However, from the halting problem, we cannot decide if a program stops or not. It is claimed that the noncomputability of Kolmogorov complexity is an example of the *Berry paradox.*

Despite the negative result that absolutely optimal compression is unknowable (i.e., researcher is never sure a particular method achieves the best compression, or the shortest program to print the data), Faloutsos and Megalooikonomou [2007] emphasize optimism from the practical point of view, there are also positive aspects. "Data Mining will always be an art, and specifically, the art for looking for better models. We are shooting for optimal (or near-optimal) compression of the given dataset, under a set of assumptions/models; the better the model, the better the compression! The big question is what are good models that we should try to fit: Gaussians, Poisson, Pareto, fractals, power laws, or something else yet to be discovered. Thus, data mining will never become boring or automated. And the search for better models will always be fun, especially in our time, that we have two wonderful coincidences: (a) unprecedented volumes of real data to look into and (b) unprecedented storage and computing power to try several models" (page 18).

Appendix B. Model Details

6.1 Use MDL approach to Detect Meta-data Anomalies

Definition 11 (Bookkeeping Example) *A database D is a collection of n journal entries where each entry has m column features (such as f_1 is effective date; f_2 is name of the approver; f_3 account debited;...). Each feature $f \in \mathcal{F}$ has a domain $dom(f)$ of possible values (e.g., there are 400 different accounts and 10 approvers). $arity(|accounts\ debited|) = 400$*

- *The domains are not necessarily distinct between features: some accounts are both debited or credited; some approvers are also initiators*
- *An item is a feature-value pair can be $\{account\text{-}debited = cash\}$;*
- *An itemset (a pattern) is a pair $\{account\text{-}debited = cash; approver = doe; ...\}$*

Step 1: Define what a model is: The model is a two-column code-table (CT)

- first column contains patterns (p), i.e., *itemsets* (F), ordered by descending by length and by support
- second column contains the codeword $code(p)$
- *usage* of $p \in CT$: number of $t \in D$ containing p in their *cover*.

CompreX exploits correlation among some features by building multiple code, probably smaller tables for each highly correlated group of features instead of a single table for all features.

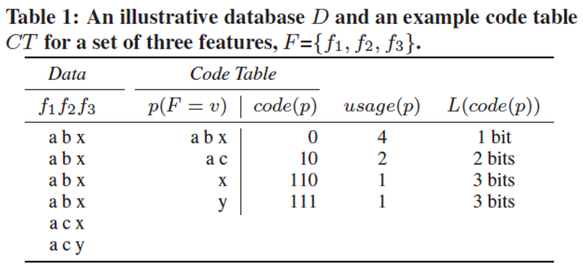


Table 1: An illustrative database D and an example code table CT for a set of three features, F=$\{f_1, f_2, f_3\}$.

Data	*Code Table*			
$f_1 f_2 f_3$	$p(F = v)$	$code(p)$	$usage(p)$	$L(code(p))$
a b x	a b x	0	4	1 bit
a b x	a c	10	2	2 bits
a b x	x	110	1	3 bits
a b x	y	111	1	3 bits
a c x				
a c y				

Figure 9: The Code Table for CompreX

Definition 12 (Feature Partitioning) *A feature partitioning $\mathcal{P} = \{F_1, F_2, ..., F_k\}$ of a set of features $\mathcal{F}$ is a collection of subsets of $\mathcal{F}$ where*

- *each subset contains one or more features: $\forall F_i \in \mathcal{P}, F_i \neq \emptyset$;*
- *all subsets are pairwise disjoint: $\forall i \neq j, F_i \cap F_j = \emptyset$; and*
- *every feature belongs to a subset: $\cup F_i = \mathcal{F}$.*

Step 3: search algorithm

- The search space for finding the best code table for a given set of features, let alone for finding the optimal partitioning of features, is quite large

  - Finding the optimal code table for a set of $|F_i|$ features involves finding all the possible patterns with different value combinations up to length $|F_i|$ and choosing a subset of those patterns that would yield the minimum total cost on the database induced on F_i.
  - Furthermore, the number of possible partitioning of a set of m features is the well-known Bell number.

- While the search space is prohibitively large, it neither has a structure nor exhibits monotonicity properties which could help us in pruning. As a result, we resort to heuristics. Our approach builds the set of code tables in a greedy bottom-up, iterative fashion.
  - Start with $\mathcal{P} = \{f_1, f_2, ..., f_m\}$
  - Calculate $IG(F_i, F_j) = H(F_i) + H(F_j) - H(F_i, F_j) = M(F_i, F_j)$ for a pair of feature-subsets of the partition
  - See Akoglu et al. [2012] for details.

6.2 Use MDL approach to Detect Graph Anomalies

Definition 13 (Bookkeeping Example) *A database $\mathcal{G}$ is a collection of J journal entries where each entry is represented as a graph $G_j = (V_j, E_j)$ with at least two nodes and one directed edge.*

- *a node $u \in V_j$ corresponds to an account such as cash or accounts receivable;*
- *a directed edge $(u, v) \in E_j$ corresponds to a credit to account u and a debit to account v;*
- *$m(u, v)$ represents the number of edges from u to v within a same journal entry G_j;*
- *$\cup_j V_j$ corresponds to the set of all accounts in the company's chart of accounts (COA)*
- *$\mathcal{T}$ denote the set of account labels: $\mathcal{T} = \{assets, liabilities, equity\}$ for example;*

Step 1: Define what a model is: The model is a two-column Motif-table ($MT \in \mathcal{MT}$)

- first column contains small graph structures, i.e., *motifs* (g), a connected, directed, node-labeled, simple graph, with possible self-loops on the nodes.
- second column contains the codeword $code_{MT}(g)$ (or c) with length $\ell(g)$

Step 3: search algorithm

Definition 14 (Formal Problem Statement) *Given a set of J node-labeled, directed, multi-graphs in $\mathcal{F}$, find a motif table $MT \in \mathcal{MT}$ such that the total compression cost in bits given below is minimized:*

$$\min_{MT \subset \mathcal{CT}} L(MT, \mathcal{G}) = L(MT) + \sum_{G_j \in \mathcal{G}} L(G_j | MT)$$

- The key idea of the motif table was to economize over frequencies of sub-graphs commonly used in lots of real journal entries (leading to patterns)

Use Compression to Detect Anomalies Compression based techniques are naturally suited for anomaly and rare instance detection. This is how we exploit the dictionary based compression framework for this task

- In a given Motif table, the patterns with short code words, that is those that have high usage, represent the sub-graphs in the graph database that can effectively compress the majority of the data points.
- Consequently, the graphs in a graph database can be scored by their encoding cost for anomalousness.
- Formally, for every tuple $t \in D$, compute the anomaly score:

$$score(G_j) = L(G_j|MT) = \sum_{g \in \mathcal{M} : g \in cover(G_j)} L(code(g)|MT)$$

- The higher the score, the the more likely it is "to arouse suspicion that it was generated by a different mechanism"

6.3 MDL for Evaluation of Account Classification

Problem Statement Given a large graph that is *node-labeled, directed, multi-graphs*, create a summary graph which representative summary that facilitates the visualization,

Definition 15 (Formal: Summary Graph) *Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a directed graph with multiplicity $m(e) \in \mathbf{N}$ and node type $l(u) \in \mathcal{T}$. A Summary graph is a graph $\mathcal{G}_s = \{\mathcal{V}_s, \mathcal{E}_s\}$ where every super node $v \in \mathcal{V}_s$ is annotated by four components*

- *$l(v) \in \mathcal{T}$ is depicted by color;*
- *$|\mathcal{S}_v|$ denote the number of nodes it contains, depicted by size;*
- *the glyph $\mu(v) \in \mathcal{M}$ depicted by shape; and*
- *the representative mulitplicity $m(v)$ of the edges it summarizes, depicted by a scalar inside the glyph.*
- *Each super edge $e \in \mathcal{E}_s$ is annotated by $m(e)$ be the representative mulitplicity of the edges between super nodes it captures, depicted by a scalar on the super edge.*

Now we define Decomposition of a given summary graph.

Definition 16 (Formal: Decompression) *A Summary graph $\mathcal{G}_s = \{\mathcal{V}_s, \mathcal{E}_s\}$ with the annotation above decompresses uniquely and unambiguously into $\mathcal{G}' = dec(\mathcal{G}_s)\{\mathcal{V}, \mathcal{E}'\}$ according to simple and intuitive rules:*

- *every super node expands to the set of nodes it contains, all of which also inherit the super-node's type;*
- *the nodes are then connected according to the super-node's glyph (for out(in)-stars a node defined as the hub points to (is pointed by) all other nodes, for cliques all possible directed edges are added between the nodes, and for disconnected sets no edges are added);*
- *super-edges expand to sets of edges that have the same direction (If the source/target glyphs involved are not stars, all nodes contained in source glyph point to all nodes contained in target glyph. For stars, expanded incoming and out-going super-edges are only connected to the star's hub); and*
- *all expanded edges obtain their corresponding "parent" super-node or super-edge representative multiplicity.*

Two-part MDL: TG-sum In summarize, to apply the MDL principle to a learning task (i.e., the summary graph), we proceed in three main steps.

- **Step 1: Define what is the Model** The model here consists of list of subsets (v's) of original nodes ($\mathcal{V}$) to merge, a list of glyphs (μ's) to design for a given graph $\mathcal{G}$
- **Step 2a: how to encoding summarization error given summary graph** define a suitable encoding scheme, designing a system to encode the patterns and to encode the data using such patterns, with prefix-free codes as basic ingredients.
- **Step 2b: how to encode summary graph based on the model** design a search algorithm, allowing to identify in the data a collection of patterns that yields a good compression under the chosen encoding scheme.
- **Step 3: search for the best model**

  Definition 17 (Formal Problem Statement) *Given a node-labeled, directed, multi-graph $\mathcal{G}$, find a summary graph $\mathcal{G}_s$ such that the encoding cost in bits given below is minimized:*

$$\mathcal{G}_s := arg \min_{\mathcal{G}'_s} L(\mathcal{G}'_s) + L(\mathcal{G}|\mathcal{H})$$

$$s.t. \;\; \mathcal{H} = dec(\mathcal{G}'_s)$$

References

Akoglu, L., Tong, H., Koutra, D., 2015. Graph based anomaly detection and description: a survey. Data mining and knowledge discovery 29, 626–688.

Akoglu, L., Tong, H., Vreeken, J., Faloutsos, C., 2012. Fast and reliable anomaly detection in categorical data, in: Proceedings of the 21st ACM international conference on Information and knowledge management, pp. 415–424.

Arya, A., Fellingham, J.C., Glover, J.C., Schroeder, D.A., Strang, G., 2000. Inferring transactions from financial statements. Contemporary Accounting Research 17, 366–385.

Arya, A., Fellingham, J.C., Mittendorf, B., Schroeder, D.A., 2004. Reconciling financial information at varied levels of aggregation. Contemporary Accounting Research 21, 303–324.

Bay, S., Kumaraswamy, K., Anderle, M.G., Kumar, R., Steier, D.M., 2006. Large scale detection of irregularities in accounting data, in: Sixth International Conference on Data Mining (ICDM'06), IEEE. pp. 75–86.

Beneish, M.D., 1997. Detecting gaap violation: Implications for assessing earnings management among firms with extreme financial performance. Journal of accounting and public policy 16, 271–309.

Beneish, M.D., 1999. The detection of earnings manipulation. Financial Analysts Journal 55, 24–36.

Berberidis, D., Liang, P.J., Akoglu, L., 2020. Tg-sum: Summarizing directed multi-type multi-graphs. under Blind Review Submission .

Brown, N.C., Crowley, R.M., Elliott, W.B., 2020. What are you saying? using topic to detect financial misreporting. Journal of Accounting Research 58, 237–291.

Butterworth, J.E., 1972. The accounting system as an information function. Journal of Accounting Research , 1–27.

Chaitin, G.J., 1969. On the length of programs for computing finite binary sequences: statistical considerations. Journal of the ACM (JACM) 16, 145–159.

Cover, T.M., 1999. Elements of information theory. John Wiley & Sons.

Dechow, P.M., Ge, W., Larson, C.R., Sloan, R.G., 2011. Predicting material accounting misstatements. Contemporary accounting research 28, 17–82.

Faloutsos, C., Megalooikonomou, V., 2007. On data mining, compression, and kolmogorov complexity. Data mining and knowledge discovery 15, 3–20.

Fellingham, J., 2018. The double entry system of accounting. Accounting, Economics, and Law: A Convivium 8.

Feltham, G.A., Ohlson, J.A., 1995. Valuation and clean surplus accounting for operating and financial activities. Contemporary accounting research 11, 689–731.

Galbrun, E., 2020. The minimum description length principle for pattern mining: A survey. arXiv preprint arXiv:2007.14009 .

Grünwald, P.D., 2004. A tutorial introduction to the minimum description length principle. arXiv preprint math/0406077 .

Grünwald, P.D., 2007. The minimum description length principle. MIT press.

Ijiri, Y., 1965. On the generalized inverse of an incidence matrix. Journal of the Society for Industrial and Applied Mathematics 13, 827–836.

Ijiri, Y., 1975. Theory of accounting measurement. 10, Amer Accounting Assn.

Ijiri, Y., 1993. The beauty of double-entry bookkeeping and its impact on the nature of accounting information. Economie Notes by Monte dei Paschi di Siena 22, 265–285.

Kolmogorov, A.N., 1965. Three approaches to the quantitative definition ofinformation'. Problems of information transmission 1, 1–7.

Liu, F.T., Ting, K.M., Zhou, Z.H., 2012a. Isolation-based anomaly detection. ACM Transactions on Knowledge Discovery from Data (TKDD) 6, 1–39.

Liu, Z., Yu, J.X., Cheng, H., 2012b. Approximate homogeneous graph summarization. Information and Media Technologies 7, 32–43.

McGlohon, M., Bay, S., Anderle, M.G., Steier, D.M., Faloutsos, C., 2009. Snare: a link analytic system for graph labeling and risk detection, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1265–1274.

Myung, I.J., Balasubramanian, V., Pitt, M.A., 2000. Counting probability distributions: Differential geometry and model selection. Proceedings of the National Academy of Sciences 97, 11170–11175.

Navlakha, S., Rastogi, R., Shrivastava, N., 2008. Graph summarization with bounded error, in: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp. 419–432.

Nguyen, H.T., Liang, P.J., Akoglu, L., 2020. Anomaly detection in large labeled multi-graph databases. arXiv preprint arXiv:2010.03600 .

Nissim, D., Penman, S.H., 2001. Ratio analysis and equity valuation: From research to practice. Review of accounting studies 6, 109–154.

Noble, C.C., Cook, D.J., 2003. Graph-based anomaly detection, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 631–636.

Pacioli, L., et al., 1994. Summa de Arithmetica geometria proportioni: et proportionalita... Paganino de paganini.

Penman, S.H., Penman, S.H., 2007. Financial statement analysis and security valuation. volume 3. McGraw-Hill New York.

Rissanen, J., 1978. Modeling by shortest data description. Automatica 14, 465–471.

Schroeder, M., 2009. Fractals, chaos, power laws: Minutes from an infinite paradise. Courier Corporation.

Sims, C.A., 1996. Macroeconomics and methodology. Journal of Economic Perspectives 10, 105–120.

Solomonoff, R.J., 1964. A formal theory of inductive inference. part i. Information and control 7, 1–22.