Mining Data Wrangling Workflows for Patterns, Reuse and Optimisation Opportunities*

Abdullah Khalid A Almasaud[1,*], Sandra Sampaio[1] and Pedro Sampaio[2]

[1]Department of Computer Science, University of Manchester, Manchester, UK
[2]Alliance Manchester Business School, University of Manchester, Manchester, UK

Abstract

Data wrangling (DW) is the complex process associated with preparation of raw data for analysis. It is typically performed as a craft in an *ad hoc* manner and its level of complexity and success is highly dependent on the data analysis task at hand, quality of input data, and skill set of the data analyst. This makes the reuse of DW pipelines difficult, forcing data analysts to often devise a new pipeline for each combination of data and analysis tasks, a process that is not only complex but also expensive, consuming between 50 to 80% of even the most experienced data analyst's time. In this paper, we investigate a number of DW pipelines in the form of workflows to find commonalities or patterns in the way DW is performed in practice, considering a multitude of data analysis tasks and data sets, devised by data analysts with varying levels of experience. We present our investigation as a methodology that, from selection of workflow sources to workflow mining techniques, describes how we dealt with the challenges of finding patterns in the way people prepare data for analysis, given the general lack of guidelines for best practices and standards. The obtained results provide insights into the most commonly used DW operations, solution patterns, redundancies and, not only optimisation opportunities, but also opportunities for reuse of experience and best practices in data engineering. We believe that the obtained insights can be useful in facilitating the construction of DW solutions to inexperienced data analysts via the reuse of patterns and best practices in DW.

Keywords

Data wrangling, workflow patterns, workflow mining, pattern reuse.

1. Introduction

Data Wrangling (DW) is regarded as a tedious and complex process, consuming 50-80% of data analysis time [1, 2]. The term, first used in 2011, describes a set of activities associated with the transformation of raw data into an asset ready for analysis. These activities typically include data profiling, formatting, integration, transformation, cleaning and outlier detection [3, 1] which are similar to those performed in the well established Extract-Transform-Load (ETL) process [4]. The practice of DW has been made possible to professionals lacking in data science and engineering skills by the availability of self-service tools that offer flexibility for custom, *ad hoc* and quick DW solutions, often through user-friendly Graphical User Interfaces (GUIs) [5, 6, 3].

While such tools may share, to some extent, common DW supporting functionality, there is a lack of standards in terms of requirements, representations and outcome in their offerings of DW operations [7]. This causes a severe burden on the user of these tools, resulting from the significant effort that needs to be made when choosing suitable tools for the task at hand, based on the user's understanding of the functionality associated with each available DW operation, and the time taken to develop a solution, often through a trial and error process.

This burden, combined with low levels of skill and experience, makes DW a tedious, complex, non-optimal and error-prone process. It results in users building different custom solutions from scratch for similar problems, rather than reusing existing ones, due to the difficult job of understanding the available solutions. As observed during our investigation, custom solutions available from popular repositories suffer from redundancies and lack of efficiency due to poor choices in operation selection, potentially impacting on pipeline execution time and quality of data analysis results.

Finding ways to help users build good quality DW pipelines without burdening them is essential to the successful adoption of DW tools. This raises questions, such as the following: given a wealth of Web-available repositories of DW pipelines, what solution patterns can be found in existing pipelines that could allow reuse of existing solutions to DW problems? Can *good quality* solutions to common DW problems be developed through the use of patterns found in existing solutions? A good quality solution can be one that minimises waste in the use of computational resources.

In search for answers to these questions, an investigation into existing DW pipelines was performed, enabling

DataPlat'23: 2nd International Workshop on Data Platform Design, Management, and Optimization, March 28, 2023, Ioannina, Greece
*Corresponding author.
abdullah.almasaud@postgrad.manchester.ac.uk (A. K. A. Almasaud); s.sampaio@manchester.ac.uk (S. Sampaio); p.sampaio@manchester.ac.uk (P. Sampaio)
0000-0002-7650-5027 (A. K. A. Almasaud); 0000-0002-3751-4182 (S. Sampaio); 0000-0002-5341-1327 (P. Sampaio)

identification of how analysts develop DW solutions in practice. In this paper, we present the methodology we used in this investigation, which combines ideas from previous work in fields, such as ETL and Software Engineering (SE), to discover patterns in DW workflows. More specifically, we collect a set of workflows to search for DW practices followed by analysts in different data domains, aiming at identifying common traits present in their DW pipelines. In this process, we conceptualise DW representations in a workflow as a way to standardise them and, through the use of well-established graph mining solutions, we identify frequently-occurring patterns of DW tasks and operations. We present usage information of identified patterns of operations and discuss how our findings can facilitate reuse of existing DW solutions and optimisation of DW pipelines for common DW problems. We believe that identification of reuse and optimisation opportunities in *ad hoc* processes, such as DW, can reduce burden and increase productivity of the professionals developing these processes, e.g., data analysts/scientists performing DW.

The rest of the paper is organised as follows: Section 2 presents a brief background and related work, the research methodology is presented in Section 3 followed by the findings and discussion in sections 4 and 5 then finally our conclusion and future work in Section 6.

2. Background and Related Work

At a time when data is often analogised to gold, diamonds [8] and oil [9], DW is predicted to become an increasingly critical process preceding data analysis, capable of determining the quality of the outcome of data analysis tasks [8]. To facilitate DW, a variety of tools have been proposed, ranging in capability and level of technical skill required from users [7, 10], e.g., from programming tools requiring high-level technical skills, such as R and Python programming, to visual tools requiring lower levels of technical expertise, a category that Talend Data preparation [11] and Trifacta Wrangler[12] fall into. Both categories of tools, however, are limited in the sophistication of mechanisms to facilitate reuse of existing solutions. This is a shortcoming that Software Engineers have worked hard to overcome and often tackle through the use of patterns found in software design, as a way of standardising and passing down knowledge/experience of expert designers to non-experts, as readily-available-for-reuse constructs[13]. While the factors motivating creation of design patterns in SE are similar to those in DW, no clear attempts to identify patterns in DW can be found in the literature. However, reuse of previous DW solutions via other means is found in previous work, such as the one by He *et al.* [14], which shows that suggestions of applicable operations can be made based on user provided examples, and the one by Sutton *et al.* [15], which is inspired by the UNIX *Diff* command. Although tool functionality for direct and automated reuse of existing solutions is not a contribution in any of these earlier research efforts, they bear evidence to the fact that solutions to common issues can be devised. The focus of this paper is on identification of design patterns in available DW pipelines, represented as workflows, considering that a significant number of them are found in this format. For that, a review of available workflow mining contributions is provided in the following.

Workflow mining focuses on business process mining using event logs for the purpose of redesign and optimisation [16, 17], which can't be used to mine patterns in data pipelines. Previous works identifying patterns used logical modelling of workflows into graphs to enable their mining, where Tosta *et al.* [18] uses a path approach to identifying patterns in scientific workflows (Figure 1), and Theodorou *et al.* [19] used Frequent Subgraph Mining (FSM) to identify frequent patterns of operations in ETL workflows. While the approaches used by both are applicable to DW workflows, [18]'s approach would lead to identifying a set of short frequent paths of operations which may lead to misleading conclusions, e.g, tasks C,D and E in the figure appear 4 times (once in each path) as opposed to once in the original representation. The approach of [19] was successful in identifying patterns in ETL, however, the repository used was created from well-formed ETL standard workflows.

Widely established ETL and DW approaches may involve similar subsets of operations and steps, however, the two present considerable differences. For example:

- ETL's main focus is on operations that perform data integration from multiple data sources into a unified schema, and data storage in a consolidated data repository (e.g., data warehouse) whilst DW involves tailored data transformations to prepare a dataset for an analytical task such as machine learning-based prediction.
- ETL involves a smaller subset of operations/data transformations applied in a quasi-sequential structure whilst DW involves a wider range of operations with potentially iterative cycles and interactive tasks (e.g., visual data profiling)
- ETL is typically performed using specialized software tools (e.g., Talend and Oracle Data Integrator) that automate the majority of the ETL process steps, whilst DW combines GUI-based tools (such as Trifacta) and programming languages such as Python and R to code user defined functions.

These differences result in different patterns to solve data problems faced by analysts, that are not traditionally common in ETL workflows, but the similarities between the two and their required outcomes is the reason for

our adaptation of ETL-based approaches to our research purposes.

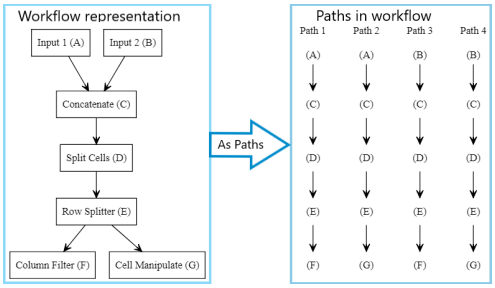

Figure 1: Interpretation of [18] approach for DW

3. Research Methodology

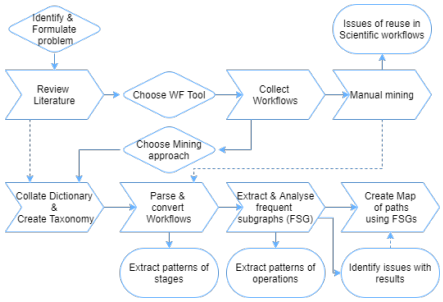

Figure 2: Research method followed in this research, diamonds represent decision points, ellipses represent findings and the remaining are steps. The solid arrows represent workflow while the dashed line represents influence.

Our investigation starts with a literature review to identify approaches to answering the research questions described in Section 1 and a search through available DW pipeline repositories for pipelines to be mined. By eyeballing through the pipelines, some initial observations were made, which helped with decisions on the approach to take and tools to use. The following sections describe each step in the methodology we used to carry out our investigation, illustrated in Figure 2.

3.1. Selection of DW Workflows

Despite the availability of numerous repositories of data manipulation workflows, the workflows found in [20, 21, 22] were narrowed-down, due to their focus on DW. Following a thorough check, the MyExperiment workflows [20] were selected for their high-level of completeness and complexity. The vast majority of the higher-quality workflows that were of interest to us in the repository were generated using the KNIME tool [23]. While the approach followed is not limited to KNIME, we chose it because it contains builtin operations covering a wide range of purposes and DW functionality, as well as the tool being reported as a leader in data science [24] and aimed at multiple data analysis applications.

Table 1
Number of Workflows From Each Repository

Source	Number of Workflows
Source 1 - myExpirement.org	76
Source 2 - NodePit	103
Source 3 - KNIME Hub	1,751
Total	**2,930**

3.2. Creation of a Taxonomy of Data Wrangling Operations

As discussed in Section 1, different tools implement DW operations in different ways, and may use the same operation name to identify different functionality, or different names to identify the same functionality. Functions implementing the same functionality may present the same name, but different function signatures in different tools. Additionally, common DW functionality may be encapsulated in a single operation in one tool, while multiple pieces of basic DW functionality may be combined as a single operation in another tool. Figure 4 provides an example of combination of common tasks into a single construct [25]. The lack of standards in DW operations makes identification of patterns in DW pipelines hard. To overcome this difficulty, a process of conceptualisation and unification of DW operations was devised and is described as follows. Considering that a data pipeline is composed of a number of steps, encompassing a number of tasks that can be executed using different operations, a dictionary of typical steps and tasks in a data pipeline can help conceptualising and unifying steps across data pipelines. Such dictionary was collated from the literature, despite the lack of a consensus on steps and tasks names, through the precise identification and concise description of DW steps and tasks obtained from Rattenbury *et al.* [26], Hellerstein *et al.* [6] and [27] were used in the development of the dictionary. The following describes the dictionary's main DW steps.

1. Loading Data (L): Obtaining the data and transforming it into a format usable by the tool.
2. Exploring Data (D): Tasks related to data discovery and validation (e.g profiling).
3. Cleaning (C): Includes two buckets of tasks: record-based cleaning (e.g removing empty records) and value-based cleaning (e.g modifying column datatypes).

4. Intra-record Structuring (S1): Transformations that manipulate individual fields or records at a time and include: removing, reordering and renaming columns as well as creating new columns.
5. Inter-record Structuring (S2): Transformations that operate over multiple records and fields at once and are divided into: filtering and reordering records, shifting granularity of dataset via aggregation or pivots, and splitting the dataset into multiple datasets.
6. Integrating Data (I): Enriching a dataset by combining it with other dataset(s) using unions or joins.
7. Enriching Data (E): Transformations that add value to the dataset by deriving new fields using equations or domain specific operations e.g deriving date of week from date.
8. Transforming Data (T): Transforming data values within fields (e.g binning or normalisation).
9. Publishing Data (P): Encompasses all tasks that indicate an end of a DW portion of a pipeline such as start of an analysis portion or exporting the dataset in a format of choice.

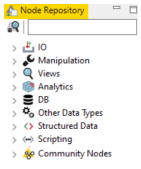

Figure 3: KNIME Node Categories

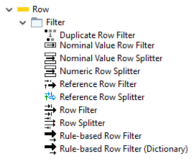

Figure 4: *Row Filter* Nodes

To build the intended taxonomy for unifying tasks in a DW pipeline, an initial set of conceptual operation labels was also created, based on the transforms presented by Raman *et al.* [28] and the Relational Algebra (RA) operations, eliminating duplicates found between the two. Further, for each label, a DW task flag was created, as presented in [28, 10, 6, 27, 26, 7, 29]. The factors summarised in Figure 5 were used to map each operation (i.e., workflow node) into a label or to create a new label for an operation, if needed, in the following order. It is worth pointing out that, the dictionary of steps helps mapping operations and labels into its most generalised form.

1. The (KNIME, in this case) node repository classification tree, illustrated in Figure 3.
2. The node's functionality (e.g., *Row Splitter* is classified as a row filter operation according to its functionality).
3. The node's description (e.g., *Substructure Search* is classified as a row filter operation since it implements domain-specific 'row split' functionality).
4. The node's prevalent trait (e.g., *Date&Time Difference* calculates the difference between two dates in a row and creates a new column with the results; as such, it can be classed under the 'append column' task).

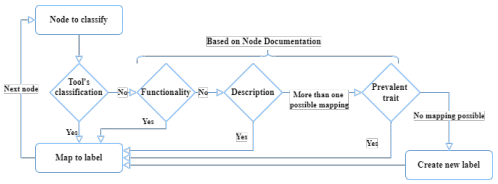

Figure 5: Node classification process flow

An example of benefits of applying the taxonomy is illustrated in Figure 6 which can also be extended to unify pipelines from multiple tools. In the example, the repository initially contains 95k nodes with 1.8k unique signatures, which got reduced to 385 signatures, by using the 90 labels in the taxonomy. The 90 created labels contain 37 DW tasks spanning 60% of the node instances.

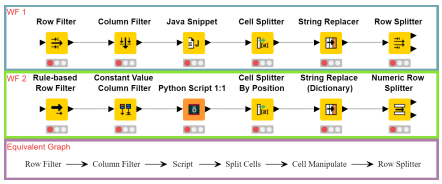

Figure 6: Two workflow snippets with unique signatures and their representation after applying the taxonomy.

3.3. Parsing of Repository Workflows into Graphs

Preparation of workflows to be parsed using the PAFI algorithm, developed by Kuramochi *et al.* [30], is summarised in Table 2. Note that, while the target workflows are, logically, direct acyclic graphs (DAG) [23], the direction of edges had to be neglected during the parsing process to adhere to PAFI's requirement of ID order in edge specification. Additionally, a concurrent parsing process was performed to produce steps graphs using the steps dictionary (described in Section 3.2). In other words, each node in every repository workflow was mapped into its step representation, where workflow duplicate step occurrences and branching were eliminated to produce a linear graph, to be used in the identification of frequent step patterns.

Table 2
Workflow Manipulations Mapped to Purpose

Group	Description	Purpose: Tool requirement	Purpose: Representation requirement	Purpose: Repository reduction
ID Related	Omit blank node IDs within workflows	✓	✓	
	Eliminate ID repetition in workflows	✓	✓	
	arrange IDs in each edge specifications in ascending order	✓		
Node	Apply the taxonomy		✓	✓
	Omit workflows with no DW operations		✓	✓
Structure	Parse each workflow into a graph and eliminate nesting	✓	✓	
	Produce graph transaction database	✓		
	Visualise the produced patterns		✓	

3.4. Mining of Workflows

The aim of the workflow mining performed in this research is twofold: *(i)* to find patterns of DW steps in pipelines, and *(ii)* to identify the frequency of DW tasks within the pipelines. For that, parsing of the workflows using the dictionary of steps, described in Section 3.2, was performed to produce a simplified graph of each pipeline while still preserving the original relevant content, as shown in Table 3.

Following transformation of DW pipelines into graphs through the application of the algorithm (and tool) proposed by Kuramochi *et al.* [30], the sets of graphs were divided into buckets based on the % of DW operations contained in them, e.g., ($\geq 80\%, \geq 60\%, \geq 40\%, \geq 20\%$and $> 0\%$), via multiple iterations performed with different support[1] *(sup)* thresholds.

3.5. Considering the Most Traversed DW Paths

The obtained results from the mining of workflows have shown that a significant fraction of the discovered patterns present low frequencies, despite the similarities they share, whose only exceptions are a few extra analysis or domain-specific operations that appear in some of them, which we informally refer to as *intruder operations*, and slight variations in the order in which DW operations appear. These observations have brought about the idea of making changes to our original approach, which considered obvious patterns appearing in the workflows (i.e., with high frequencies), as suggested by Theodorou *et al.* [19]. The modified approach aims at identifying lower frequency patterns found in the mining results that could potentially represent high frequency ones, if exceptions are dealt with differently. This approach is detailed in the next sections but, in essence, variations in operation ordering found in operation groupings that appear with a certain frequency are disregarded, allowing identification of paths of DW operations within these groupings that are frequently traversed, which we call *the Most-commonly Traversed Paths (MTPs).*

[1] the minimum proportion of workflows that should contain the subgraph to be counted as frequent

Table 3
Modifications to steps representation of workflows and purpose

Description	Purpose
Removal of User Defined Function (UDF) vertices because they require individual analysis	Simplification of graphs
Removal of duplicate step vertices both when done in sequence or not	Simplification of graphs
Removal of order of step execution, by sorting remaining vertices alphabetically	Aid in identifying patterns
Discard workflows not containing more than one of the core wrangling steps (2-8) because they wouldn't provide useful insight	database reduction
Combined both Analysis steps and output production into a Publish step because in terms of DW they mean the pipeline is over	Same results

4. Findings

Findings in this research can be categorised as shown in Table 4 based on their source as observations and mining. The first are related to reuse in workflows which were recorded while gathering the workflows and setting up the environment. The mining tasks performed, resulted in different sets of findings, which when combined provide a greater understanding of DW pipelines in a workflow setting.

The findings from observations were made while setting up the KNIME environment to work with pre-built scientific workflows. The observations made were used to identify issues related to reuse in workflows as well as identify issues that would hinder the subgraph mining process. The domain specific file structures used in sci-

Table 4
Categorised Summary of Findings

Category	Brief	Source
Reuse in scientific workflows	• Issues related to input files • Issues in preparing environment • Issues in acquiring community nodes • High use of UDFs	Observation
Frequent steps	• Order variation - high • Intertwined steps of DW and analysis • Patterns of single file DW • Patterns of multi file DW	Mining Steps
Frequent operations	• Low support values • Less authors = more patterns • High number of order permutations • Different types of repetition • Wrong/wasteful usage of operations	Mining Subgraphs
Most traversed paths	• Increase frequency of subgraphs • Unveil new significant frequent patterns • Unveil room for optimisation	Map of frequent subgraphs

entific workflows which are not natively supported by DW tools introduced the need of community to develop custom nodes that are not always updated to support the latest releases of KNIME. To solve issues related to lack of support to community developed nodes, users seem to prefer using UDFs in the form of scripts and code snippets. While reviewing these UDFs, it was noted that the majority of them perform domain specific functions that are highly idiosyncratic and with limited impact on the approach of identifying general DW patterns. The use of community developed nodes as well as the UDFs sometimes dictate the placement of specific nodes which were major contributors to issues identified in varied order of operation placements.

In the DW steps pattern mining, 31% of pipelines (out of the 1,787 pipelines investigated) were omitted because they included less than two DW steps. The remaining pipelines resulted in multiple usage patterns from which the 7 most interesting ones are presented in Figure 7. The presented patterns are divided into two groups based on their inputs a) single-input patterns (*Ps1-3*) and b) multi-input patterns (*Pm1-4*). The most frequent occurring pattern *Pm3* was implemented in 32% of the workflow instances followed by 16% for *Ps2*. The step patterns are implemented by users based on factors such as number of inputs, structure of input(s) and subsequent analysis requirement as seen in Table 5, e.g: if one has a dataset requiring dealing with null values (imputation or removal) and column based operations (e.g creation of new columns) they would use pattern *Ps1*. The results of DW steps mining confirmed that the lack of rules on operation placement was not simply an interchange of adjacent tasks which would not appear if generalised but was a more critical issue that appeared even when generalised to the level of steps. An example of the high variation of order can be seen in *Pm3* which has 5 core wrangling steps and appeared in 200 different order permutations.

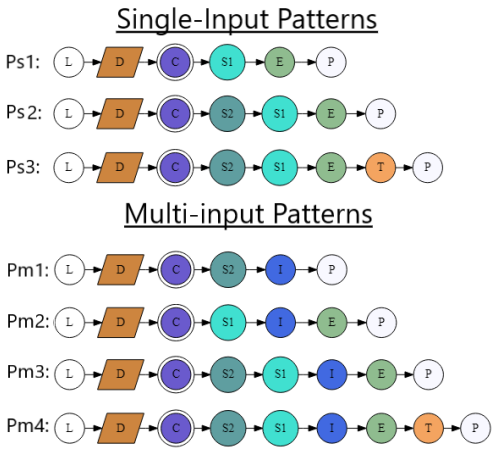

Figure 7: Most interesting patterns, parallelogram represent Exploration (D) which is not always captured by workflows and the double circle used because the cleaning (C) step is optional and data dependant.

The highest *sup* value with results in the task level pattern mining was 50% when the database of *Source1* was used which is the group that resulted in the largest number of patterns in other *sup* values. The highest *sup* value producing patterns for all databases was 15% as illustrated in Figure 8. The option to retrieve maximal subgraphs[2] only was used when running the algorithm, however, the inconsistent order of operations required further manual processing to remove duplicate patterns and those never independently appearing in workflows. The DW tasks were present in 575 subgraphs of which 63% contained repetition of tasks (Table 6) and 50% of the remaining subgraphs consisted of 6 tasks (not all DW tasks). Although the produced subgraphs were unique to the graph mining algorithm, they were not unique in terms of functionality when analysed visually.

We define a DW pattern as a frequently occurring combination of DW tasks with distinguishing behaviour leading to its classification. The 15 most interesting patterns extracted are presented in Figure 9. Other patterns

[2]A subgraph for which none of its super-graphs is frequent.

Table 5
Pattern Descriptions Sorted by Percentage of Occurrences, P is the pattern code, WFs is the number of workflows and % is the percentage from total workflows

P	Description/Usage	WFs	%
Pm3	Dealing with missing values, dimension manipulation, shifting the granularity (aggregation or pivot) and reshaping (transpose) to prepare the data for integration.	572	32
Ps2	Dealing with missing values, dimension manipulation, shifting the granularity and reshaping.	292	16
Pm4	Same as *Pm3* but with transformation of values which is usually linked to certain machine learning use-cases.	121	7
Pm1	Dealing with missing values, dimension manipulation, shifting the granularity and reshaping to prepare the data for integration.	79	4
Ps1	Dealing with missing values and dimension manipulation.	47	3
Pm2	Dealing with missing values, dimension manipulation to prepare the data for integration and enrichment of dataset.	29	2
Ps3	Same as *Ps2* but with transformation of values which is usually linked to certain machine learning use-cases.	17	1

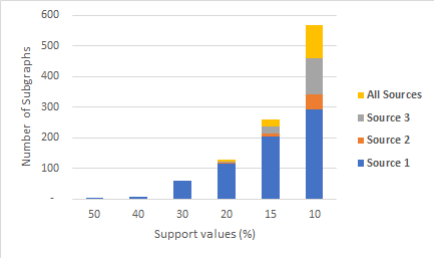

Figure 8: Results of Mining graph databases with different *sup* values

in the results were either subsumed within the 15 patterns or have no clear distinguishing behaviour that could lead to its definitive classification. Issues faced in the mining of patterns was mainly resulting from the varied order of tasks as well as the intruding tasks appearing in frequent combinations which hinder the identification of a pattern or results in them appearing with a low frequency. From the results, there were interesting insights that can be confirmed with the appearance of highly repeated operations (Table 7) which as well as producing irrelevant frequent subgraphs, indicated issues related to incorrect utilisation of operations, e.g using a RS node

Table 6
Breakdown of Types of Repetition in Subgraphs

Type of Repetition (name)	# of graphs	% from repetition	overall %
same operation (self-loop)	183	50	32
sequence of operations (SameSeqOps)	64	17	11
both types (Both repetitions)	119	33	21

Table 7
Most Repeated Operations in a Single Subgraph and Number of Graphs They Appear in

Repeated Operation	Num. Graphs	% of results
Row filter	214	37
Split Rows	52	9
Concatenate	19	3
Column Convert	17	3

while only processing a single output branch which could be achieved with a simple RF operation. Combining the subgraphs as paths to create the DW traversed paths by removing the repetition of operations and disregarding the order of their appearance in the frequent subgraphs unveiled potential significant DW patterns. Figure 10 represents one of the MTPs departing from Row Filter which contain significant insights which didn't appear in the produced subgraphs. The figure contains 5 nodes, the *path A* appeared in 45 different subgraphs, the *path B* appeared in 44 and the *path C* appeared in 27. While the number of times these paths of operations appear is relatively high, they were missed by the mining algorithm or returned with lower frequency because of the various placement order and repetition. Considering the results from the MTPs identified, lower frequency patterns in Figure 9 should have been produced by the mining algorithm with higher *sup* values had a consistent arrangement of operations been followed, e.g *path B* represent the patterns "Join with summary" and "join Summaries". On the other hand, *path C* did not clearly appear in the patterns resulting from the mining although it represents a relatively valid scenario in DW of shifting shape of a summary table before integrating datasets.

5. Discussion

The findings related to reuse of scientific workflows combined with general challenges of reuse in workflows add to the burdens faced by users. While tools such as KNIME attempt to develop nodes based on usage statistics and user feedback, this doesn't address reuse of existing workflows since users are required to fully understand a workflow to be able to modify it. The heavy use of UDFs, can be attributed to users reusing existing code to per-

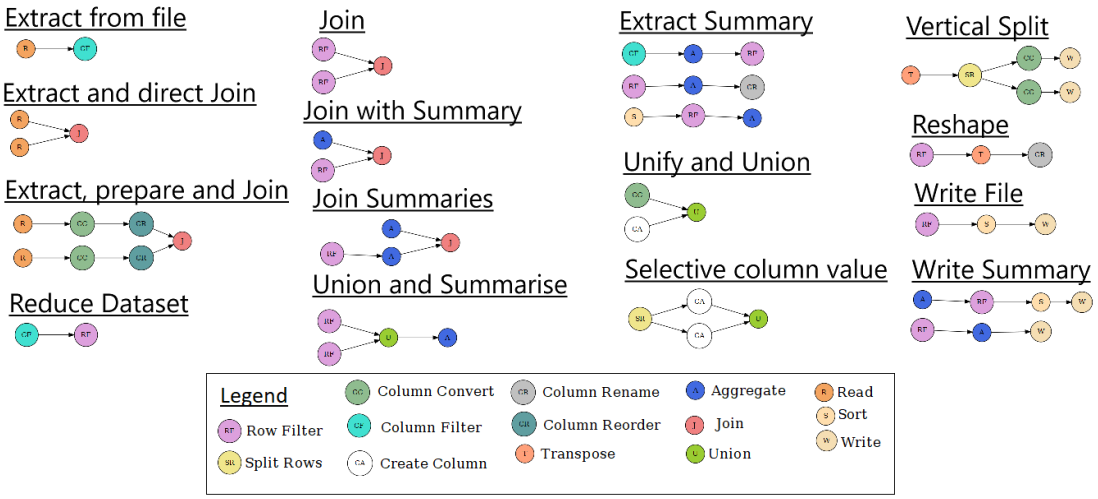

Figure 9: Most interesting patterns

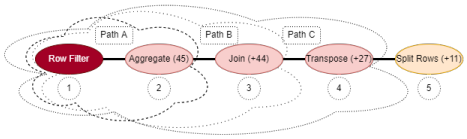

Figure 10: MTP in the map of paths departing from RF

form common tasks and in the most extreme scenarios use these interactive workflow design tools merely for their orchestration capability.

Issues of placement of operations resulting in multiple order permutations of DW steps as well as tasks can be seen in all types of data-centric workflows implemented using interactive tools can be attributed to the users and tools. The lack of technical knowledge and DW experience of the wider population of non-technical analysts combined with the usage nature of interactive tool can be attributed to poor design outcomes in investigated DW pipelines. The interactive tool design tempts a user to only use an operation to fulfil the requirement of a subsequent operation including but not limited to loading and wrangling additional dataset(s) for integration. Although the patterns presented in Figure 7 had numerous permutations during discovery, the order presented in the figure is influenced by rules of RA where operations with lower-cost and highest impact on reducing size of the data are applied as early as possible and before more expensive operations such as integration.

With regard to DW patterns, it was observed that the highest number of patterns appeared in the databases of "source1" which can be attributed to having the least number of workflows that were authored by a small group of users analysing data in similar domains. Additionally, the operations in (Table 7), indicate lack of planning as well as knowledge of the tool's capabilities. Although in some cases it may be valid to have a sequence of operations e.g to do selective processing, it is hard to find an argument which makes the sequence of multiple RF operations valid. Since RF operations evaluate a predicate by performing a full scan of the dataset, its repetition indicates lack of knowledge and/or skill to create a proper predicate even with GUI support, which results in bad workflow designs and unnecessary execution of operations. This in addition to the improper use of constructs mentioned earlier in the case of operations such as RS.

The workflows included in the repository are not purely DW workflows, nor were they necessarily created by experienced wranglers or ETL engineers, they represent a subset of real data pipelines containing DW steps in a workflow setting. While the subset of DW operations used is not big, identical combinations of operations were not produced by the mining algorithm indicating the lack of rules of thumb in DW which also explains the lack of significant highly frequent patterns of operations similar to those found in the works of [19] where the mining was performed on ETL workflows built by ETL experts using specification of TPC-DI, a benchmark for Data Integration [31]. It can also be deduced that even though a visual interface makes it easier for non-experts to attempt to wrangle by making the operations easier to find through different interfaces it doesn't guarantee they will be used effectively or efficiently without standards or design rules to follow. The findings and deductions made, resemble the basis of the motivation of the works

of the Gang of Four (GoF) in SE when they started the creation of their patterns catalogue [13], however, the difference in mentality of the two sets of users must be considered. On the other hand, the set of paths identified from the mining, present solutions that solve a wide set of wrangling use-cases such as filtering, integration, structure transformations and value conversions. E.g Fig. 10 includes very expensive operations in terms of time, memory and CPU resources that are used in varied order in practice but can be optimally arranged using query optimisation techniques to produce a semantically equivalent solution using the same operations, which is efficient to run and easy to follow and apply. Using design principles from SE and applying them to basic DW operations and MTPs would create the standards and patterns DW lacks which would be a seed to a catalogue to support the design of DW solutions.

5.1. Recommendations

Other Data engineering (DE) approaches such as query builders (QB) applied to support the creation of SQL statements without the burden of learning a new language using smart GUIs can reap benefits to building DW pipelines efficiently. These approaches have been applied in commercial ETL tools, and considering the similarities between DW and ETL, they can be used to facilitate effective DW solution design. While in ETL, simpler processes are handled using SQL [4], which can be applied in DW with the current adoption of representing data as relations in DW tools, it wouldn't reduce burdens of DW on non-technical users. This adopted representation combined with standardisation of DW operations would result in the applicability of query optimisation techniques in DW pipelines which can then be utilised by the DW pipeline builders -similar to QB- to produce an optimised pipeline which utilises the patterns in the form of design principles. The burden of building pipelines can be further reduced by using standard statistics of input datasets to provide recommendations and solutions to a set of issues such as data reduction and integration. In data lakes, integrating metadata handling frameworks, e.g MOSES [32], would assist wranglers in identifying relevant datasets and availing the metadata to the QB without burdening the user. These approaches would result in increasing reuse of pipeline strategies rather than the complete workflows by promoting a systematic use of heuristics as well as scientific formulation of DW tasks to transform DW into an engineering discipline[33].

6. Conclusion and Future Work

While some may argue ETL and DW are fundamentally different [6], others view ETL as a type of DW overseen by specialised personnel in an organisation [26]. We agree with the latter and believe that an ETL pipeline is a well-defined DW pipeline constructed for reuse. The well-defined practices in ETL were aided by many related IT fields including adoption of techniques from SE and DE domains as well as relying on DB technologies and its optimisations when performing simple tasks.

The self-service DW tools adopting GUIs and menu-driven tool along with a spreadsheet approach have reduced the burden of programming DW recipes by using familiar interfaces, however, they do not reduce all the burdens of building a DW pipeline and in times they increase the difficulties by overwhelming users with choices of DW operation with different implementations that increases the search space of possible operations to be used by the non-technical analysts. The clear lack of standard in implementations of DW operations, their requirements and outcomes also adds an overhead to user adoption to new tools. We believe standardising operations and patterns in the field of DW and building on works such as [28] combined with clear cost models would enable optimisations to be carried in any DW pipeline. Our future work would use findings in this research to standardise the set of commonly used operations building on the idea of programming language and tool agnostic conceptual DW operations [34], formal DW pattern specification, and optimised MTPs using design principles enabling the creation of tools that reduce the burden of building DW pipelines.

References

[1] N. Paton, Automating data preparation: Can we? should we? must we?, in: Proceedings of the 21st International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data, 2019.

[2] S. M. F. Ali, R. Wrembel, Framework to optimize data processing pipelines using performance metrics, in: International Conference on Big Data Analytics and Knowledge Discovery, Springer, 2020, pp. 131–140.

[3] S. Kandel, A. Paepcke, J. Hellerstein, J. Heer, Wrangler: Interactive visual specification of data transformation scripts, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 3363–3372.

[4] S. M. F. Ali, R. Wrembel, From conceptual design to performance optimization of etl workflows: current state of research and open problems, The VLDB Journal 26 (2017) 777–801.

[5] P. Howard, Data preparation (self-service) (technology) - bloor research, 2018.

[6] J. M. Hellerstein, J. Heer, S. Kandel, Self-service data preparation: Research to practice., IEEE Data Eng. Bull. 41 (2018) 23–34.

[7] M. Hameed, F. Naumann, Data preparation: A survey of commercial tools, ACM SIGMOD Record 49 (2020) 18–29.

[8] M. Muller, I. Lange, D. Wang, D. Piorkowski, J. Tsay, Q. V. Liao, C. Dugan, T. Erickson, How data science workers work with data: Discovery, capture, curation, design, creation, in: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 1–15.

[9] W. M. P. van der Aalst, Data scientist: The engineer of the future, in: K. Mertins, F. Bénaben, R. Poler, J.-P. Bourrières (Eds.), Enterprise Interoperability VI, Springer International Publishing, Cham, 2014, pp. 13–26.

[10] G. Convertino, A. Echenique, Self-service data preparation and analysis by business users: New needs, skills, and tools, in: Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems, 2017, pp. 1075–1083.

[11] Talend, Talend data preparation, https://talend.com/products/data-preparation/, 2021.

[12] Trifacta, Trifacta wrangler, https://trifacta.com/, 2021.

[13] E. Gamma, R. Helm, R. Johnson, J. Vlissides, D. Patterns, Elements of reusable object-oriented software, Design Patterns. massachusetts: Addison-Wesley Publishing Company (1995).

[14] Y. He, X. Chu, K. Ganjam, Y. Zheng, V. Narasayya, S. Chaudhuri, Transform-data-by-example (tde): An extensible search engine for data transformations, Proc. VLDB Endow. 11 (2018) 1165–1177.

[15] C. Sutton, T. Hobson, J. Geddes, R. Caruana, Data diff: Interpretable, executable summaries of changes in distributions for data wrangling, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 2279–2288.

[16] W. van der Aalst, B. van Dongen, J. Herbst, L. Maruster, G. Schimm, A. Weijters, Workflow mining: A survey of issues and approaches, Data & Knowledge Engineering 47 (2003) 237–267.

[17] M. Hammori, J. Herbst, N. Kleiner, Interactive workflow mining—requirements, concepts and implementation, Data & Knowledge Engineering 56 (2006) 41–63. Business Process Management.

[18] F. E. Tosta, V. Braganholo, L. Murta, M. Mattoso, Improving workflow design by mining reusable tasks, Journal of the Brazilian Computer Society 21 (2015) 1–16.

[19] V. Theodorou, A. Abelló, M. Thiele, W. Lehner, Frequent patterns in etl workflows: An empirical approach, Data & Knowledge Engineering 112 (2017) 1–16.

[20] D. De Roure, C. Goble, R. Stevens, The design and realisation of the myexperiment virtual research environment for social sharing of workflows, Future Generation Computer Systems 25 (2009) 561–567.

[21] NodePit, Nodepit, https://NodePit.com/, 2021.

[22] KNIME, Knime-hub, https://hub.knime.com/, 2021.

[23] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, B. Wiswedel, Knime: The konstanz information miner, in: Data Analysis , Machine Learning and Applications : Proceedings of the 31st Annual Conference of the Gesellschaft für Klassifikation e. V., Albert-Ludwigs-Universität Freiburg, March 7 9 , 2007, Springer, New York, 2007.

[24] KNIME.com, Knime recognized by gartner as a leader in data science and machine learning platforms, https://knime.com/about/news/knime-recognized-by-gartner-as-a-leader-in-data-science-and-machine-learning-platforms-2019, 2019. Accessed Dec 1, 2019.

[25] N. Weskamp, Workflow mining, in: Encyclopedia of Database Systems, 2009.

[26] T. Rattenbury, J. M. Hellerstein, J. Heer, S. Kandel, C. Carreras, Principles of data wrangling: Practical techniques for data preparation, " O'Reilly Media, Inc.", 2017.

[27] data wrangling handbook 0.1 documentation, 2013. Accessed Dec 1, 2019.

[28] V. Raman, J. M. Hellerstein, Potter's wheel: An interactive data cleaning system, in: VLDB, volume 1, 2001, pp. 381–390.

[29] O. Azeroual, Data wrangling in database systems: Purging of dirty data, Data 5 (2020) 50.

[30] M. Kuramochi, G. Karypis, An efficient algorithm for discovering frequent subgraphs, IEEE transactions on Knowledge and Data Engineering 16 (2004) 1038–1051.

[31] M. Poess, T. Rabl, H.-A. Jacobsen, B. Caufield, Tpc-di: The first industry benchmark for data integration, Proc. VLDB Endow. 7 (2014) 1367–1378.

[32] M. Francia, E. Gallinucci, M. Golfarelli, A. G. Leoni, S. Rizzi, N. Santolini, Making data platforms smarter with moses, Future Generation Computer Systems 125 (2021) 299–313.

[33] R. Mall, Fundamentals of Software Engineering, 4th ed., Prentice-Hall of India Pvt.Ltd, 2014.

[34] S. Sampaio, M. Aljubairah, H. A. Permana, P. Sampaio, A conceptual approach for supporting traffic data wrangling tasks, The Computer Journal 62 (2019) 461–480.