Abstract: The role of data mining is to search “the space of candidate hypotheses” to offer solutions, whereas the role of statistics is to validate the hypotheses offered by the data-mining process. In this paper we propose Association Rules Networks (ARNs) as a structure for synthesizing, pruning, and analyzing a collection of association rules to construct candidate hypotheses. From a knowledge discovery perspective, ARNs allow for a goal-centric, context-driven analysis of the output of association rules algorithms. From a mathematical perspective, ARNs are instances of backward-directed hypergraphs. Using two extensive case studies, we show how ARNs and statistical theory can be combined to generate and test hypotheses.

Keywords: data mining; association rules; association rules network

1. INTRODUCTION AND MOTIVATION

Data mining is often described as the process of discovering “interesting and actionable” patterns in large databases [1]. Given the large quantity of digital data that is constantly being generated and stored, data mining offers a solution to the problem of rapidly summarizing and searching for nonobvious relationships in large databases. The role of data mining can be perceived as a methodology for discovering candidate hypothesis or theories, whereas statistical methods can be used to validate the proposed theories against data. This augmentation allows us to exploit the recent advances in computationally efficient search techniques designed for large databases in conjunction with traditional statistical methods, which remain the bedrock of theory verification and validation.

Figure 1(a) depicts a traditional theory-building process. The starting point of such a process is observations (events) that trigger the researcher to make a conceptual leap, and arrives at a framework in which the structure of the underlying process (that is generating the events) can be elicited. In order to make the theory as unambiguous as possible, a mathematical model to represent the theory is often constructed. The theory is validated by testing the mathematical model and the hypotheses suggested by it, using appropriate data. Figure 1(b) shows how the traditional approach can be augmented with data-mining techniques. For example, consider the classical data-mining technique known as association rule mining (ARM) [1,3–5]. The objective of ARM is to find “interesting patterns” in the form of rules \(X \rightarrow Y\) where \(X\) and \(Y\) can be attributes, items or more generally “data objects”. Now, if we knew beforehand that \(X\) and \(Y\) are correlated, in a statistical sense, then the discovery of the rule \(X \rightarrow Y\) does not necessarily add any useful information. On the other hand if \(X\) and \(Y\) were previously never tested for correlation, then the discovery of the rule \(X \rightarrow Y\) suggests that \(X\) and \(Y\) are candidate pairs to be validated statistically (for correlation). As datasets are generally increasing, both in quantity and dimensionality, enumerating all possible combinations of \(X\) and \(Y\) and then checking them for correlation is not computationally feasible. Thus data mining can be seen as a mechanism to offer candidate theories which are then required to be validated by statistical approaches. In the economics literature this approach is sometimes, perhaps derisively, called “measurement without theory” [2]. The argument is that without an underlying theory (called the maintained hypothesis), measurements may lead to results that are patently spurious. A well-known example is the observed positive correlation between the number of stork nests and human babies born during spring time in Scandinavian countries [6]. However, to

Correspondence to: Sanjay Chawla (chawla@it.usyd.edu.au)
conclude and explain that the relationship is spurious (and not causal) requires knowledge that cannot be inferred from the data alone. This is sometimes also called the “data-mining trap”.

However, it is time to revisit this debate given the tremendous computational advances that have been made in recent years. Research in data-mining algorithms [1,5] has now made it possible to efficiently sift through large quantities of data and generate candidate patterns which are “interesting”, often surprising and in many instances more credible than the storks/babies example would appear to suggest. In this paper, we present two examples of how the steps shown in Fig. 1 can be illustrated.

One of the challenges of data-mining techniques is that they often generate a large number of “patterns” or hypotheses, making it extremely hard for a researcher to decide which hypotheses are credible and worth evaluating using statistical techniques. What we are proposing is a method [association rules network (ARN)] to overcome this challenge for association rules—a well-known data-mining technique [1,3–5].

The core idea in ARN is that rules discovered by the association rule-mining algorithm can be synthesized, pruned, and integrated in the context of specific objectives. In particular, if there is a variable of interest (the “goal”), then we can form a network consisting of the relevant and related variables and use the network to inform a “model” that can be tested using statistical methods. For example, suppose an association rule task outputs the following rules: \( A \rightarrow B, B \rightarrow C \) and \( D \rightarrow C \). Also, the variable of interest may be \( C \). The output of the association rule task immediately suggests the regression model \( C = \alpha_1D + \alpha_2B + \epsilon \). Furthermore, we can also test for the indirect dependence of \( C \) by \( A \) and even the joint dependence of \( B \) and \( D \) on \( C \). This way we can couple a data-mining task with statistical analysis. In practice, ARM involves hundreds, if not thousands, of variables. The pruning strategy that accompanies Association Rules Networks (ARNs) can be used to remove local inconsistencies between variables (like cycles), in order to suggest consistent statistical models. To summarize, ARNs offer the following features:

1. **Pruning in context**: We use the ARN to prune rules in the context of a specific objective. Changing the objective will result in the pruning of different rules. This dynamic pruning strategy for association rules has, to the best of our knowledge, never been proposed before.
2. **Network structure**: The ARN provides a mechanism for determining the network relationship between the relevant variables and the goal. This can help us elicit direct and indirect and join effects from the network.
3. **Hypotheses generation for evaluation**: ARN can serve as a bridge between the outputs generated by ARM and their statistical evaluation.

Fig. 1 Comparison of theory driven and augmented approach. Panel (a) is the classical method of scientific inquiry [2].

---

1 Patterns discovered in the data that have no predictive value.
The rest of the paper is organized as follows. In Section 2, we provide an overview of ARM and related research. Details of ARN, such as its definition, construction algorithm, and properties, are presented in Section 3. In Section 4, we describe two comprehensive case studies that demonstrate the process of constructing and using ARNs. We also provide a validation of our results by using classical statistical methods. We conclude with Section 5 comprising a summary and directions for future work. Preliminary versions of the paper highlighting different aspects of ARN have appeared in [7,8].

2. ASSOCIATION RULE MINING AND RELATED RESEARCH

Association rule mining (ARM) is considered a cornerstone of data-mining research [3,5]. The objective of this technique is to find dependencies between different variables (called items) in a large database. Unlike a standard statistical approach to test for correlation between variables, ARM casts the problem as a search for dependencies (called association rules). In order to scale up the method for very large databases and variables, two measures (called support and confidence) are used to guide the search process.

2.1. Association Rules

Association rules are generally described in the framework of market basket analysis. Given a set of items \( I \) and a set of transactions \( T \) consisting of subsets of \( I \), an association rule is a relationship of the form \( A \rightarrow B \), where \( A \) and \( B \) are subsets of \( I \), while \( s \) and \( c \) are the support and confidence of the rule. \( A \) is called the antecedent and \( B \) the consequent of the rule. The support \( \sigma(A) \) of a subset \( A \) of \( I \) is defined as the percentage of transactions which contain \( A \) and the confidence of a rule \( A \rightarrow B \) is \( \frac{\sigma(A \cup B)}{\sigma(A)} \). Most algorithms for association rule discovery take advantage of the antimonotonicity property exhibited by the support level: If \( A \subset B \) then \( \sigma(A) \geq \sigma(B) \).

In the language of ARM, data is a set of transactions and each transaction is a subset of the item space. Our focus is on applying ARN to relational data, where each instance is a tuple which consists of a set of attribute-value pairs. More specifically, suppose we are given a relation \( R(A_1, A_2, \ldots, A_n) \) where the domain of \( A_i \) dom\( (A_i) = \{a_{i1}, \ldots, a_{in}\} \), is discrete-valued. Then an item is an attribute-value pair \( [A_i = a] \). The ARN will be constructed using rules of the form \( \{A_m = a_m, \ldots, A_n = a_n\} \rightarrow \{A_j = a\} \) where \( j \not\in \{m_1, \ldots, m_k\} \).

2.2. Association-Rules-Related Research

The earlier research in ARM has focused on algorithms for discovery [3,4] and subsequently on pruning [9,10]. This work is motivated by the fact that the standard measures of support and confidence (described above) generate many redundant rules, some of which may be spurious. Our study is also motivated by developing a solution to the problem of organizing this large number of rules so as to study a specific goal. Thus, to provide a context for the utility of ARNs, we provide a review of the literature on pruning a large set of association rules below.

In general, there are three approaches for pruning redundant rules. The first approach involves the extraction of more specific types of patterns from transaction data. Such patterns include closed itemsets [11–13], which are maximal itemsets all of whose subsets have the same support, hypercliques [14], which eliminate cross-support patterns using the h-confidence measure, and correlated patterns [15], which are sets of correlated items derived on the basis of their mutual information. Generally, these patterns are a subset of the entire set of frequent itemsets derived from a dataset.

The second approach is based on the satisfaction of an additional interestingness measure in addition to support and confidence. For example, Piatetsky–Shapiro [16] proposed that a rule \( A \rightarrow B \) is interesting if \( \frac{P(A \cup B)}{P(A)P(B)} > 1 \). Several other statistical measures of interestingness have been proposed, and Tan et al. [17] and Geng et al. [18] provide excellent surveys of the properties of the different measures. Recently, some approaches have also been proposed to determine the interestingness of a set of frequent itemsets using background knowledge in the form of prior beliefs of the user [19] or Bayesian networks representing the causal structure in the input dataset [20]. Usually, these approaches are quite successful in pruning the set of frequent item sets, and rules derived from them. Interestingly, ARNs can still be used in combination with these patterns and interestingness measures, since the ARN construction algorithm presumes only a set of rules as input, and is independent of how these rules are derived. Furthermore, these patterns and measures do not perform a goal-driven pruning, which is the main motivation behind the construction of an ARN.

The third approach, which is most relevant to our work, involves the summarization of the set of item sets and/or association rules derived from a dataset. Some techniques have been proposed for summarizing the set of frequent itemsets using probabilistic methods [21,22]. Some techniques have also been proposed for clustering frequent item sets [23,24] and association rules [25] using suitable distance measures. In [23], Han et al. proposed a method of summarizing association rules and clustering the rule items using an undirected hypergraph. Each frequent itemset was
represented as a hyperedge and set of all the frequent itemsets as a hypergraph. The resulting hypergraph was partitioned into clusters using a min-cut algorithm specifically designed for hypergraphs. Items which belonged to the same cluster were considered to be “more related” than items belonging to different clusters.

Finally, Berrado and Runger [26] have recently proposed a mechanism of generating meta-rules from a collection of discovered association rules. Here, meta-rules are generated by reapplying the ARM algorithm on the discovered rules; more precisely, on rules with the same consequent. These techniques are generally able to extract useful summaries from the large set of rules or item sets derived from a dataset, but usually involve some loss of information, since they are not able to retain the complete information in the original rule set. Furthermore, in contrast to ARN, the summaries, clusters, or meta-rules discovered are global and do not take into consideration the mining task, i.e. the goal being investigated.

In summary, although a substantial amount of work has been done in the direction of pruning the usually large set of frequent patterns or association rules derived from a dataset, ARNs represent a new dimension in this direction, since the pruning is done in the context of a goal node. This is relevant for several applications where a specific goal is being studied, such as in the case studies presented below.

3. ASSOCIATION RULES NETWORK (ARN)

Although an algorithm like Apriori will discover all association rules which satisfy the minimum support and minimum confidence thresholds, there is no guarantee that the rules discovered are meaningful or not spurious. In fact, one of the major weaknesses of this approach is precisely that several rules, many of them redundant and obvious, are produced by the algorithm. This can often hamper the knowledge discovery process.

However, in practice, the data-mining process is rarely carried out in isolation without any reference to specific user goals or focus on target items of interest. This calls for an interactive strategy by which pruning of association rules is carried out in the context of precise goals as the following example will make clear. We begin with the following rule discovered from the Census Data of Elderly People [10], which captures information about the demographic and income distribution of elderly people in the United States of America. Our goal is to understand the itemsets that are frequently associated with the target item, “income=below 50K”.²

² All rules have predefined minimum support and confidence, but the exact value is not relevant to the discussion as what we trying to highlight can occur at any support level.

\[ r_1 : \text{immigrant} = \text{no} \rightarrow \text{income} = \text{below 50k}. \]

By itself, this rule appears to contradict our general perception regarding the comparative incomes of immigrants and nonimmigrants, at least in the USA. However, combining this pattern with another rule,

\[ r_2 : \text{sex} = \text{female} \land \text{age} < 75 \rightarrow \text{immigrant} = \text{no}, \]

provides a context which helps in interpreting the first rule. This forms a network which flows into the goal (income = below50k) and provides a better explanation of the goal as opposed to when the rule \( r_1 \) is viewed in isolation. Now consider a third rule

\[ r_3 : \text{immigrant} = \text{no} \rightarrow \text{sex} = \text{female}. \]

This rule appears to be flowing “in the opposite direction”, i.e. it is redundant and does not help explain the goal. Graphically this rule participates in a hypercycle with rule \( r_2 \). Finally, consider two more rules

\[ r_4 : \text{urban} = \text{no} \rightarrow \text{income} = \text{below50k}, \]

\[ r_5 : \text{urban} = \text{no} \rightarrow \text{sex} = \text{female}. \]

The rule \( r_5 \) in conjunction with rules \( r_2 \) and \( r_1 \) also appears to be leading indirectly (redundant path) to the goal. By ignoring rule \( r_5 \), the goal items remain reachable from the item urban = no. Figure 2 captures the preceding discussion. The directed hypergraph after the removal of hyperedges \( r_3 \) and \( r_5 \), which are redundant in the presence of other rules, as shown in Fig. 2, is called an ARM. We introduced ARNs in the context of a specific application in [7], and provided a brief introduction in [8]. Here we develop this concept further in its full generality.

Our approach is based on formalizing the intuition behind these observations. It consists of mapping a set of association rules into a directed hypergraph,³ and systematically removing circular paths and redundant and backward hyperedges that may obscure the relationship between the target and other frequent items. This offers the following advantages:

(1) The pruning process is adaptive with respect to the choice of the goal node made by the user. Thus pruning is carried out with respect to the target objective. Different objectives will result in different rules being pruned.

³ Background material on directed hypergraphs [27] can be found in Section 3.2.
Fig. 2 A backward-directed hypergraph (B-graph) representing the rules r1–r5. After the removal of rules r3 and r5, this graph is called an ARN.

(2) Pruning is reduced to hypercycle and reverse hyperedge elimination in the hypergraph. This step can be readily automated (see ARN construction algorithm).

(3) The resulting hypergraph can be transformed into a reasoning network where all edges move “forward” toward the goal node. A path from a node in the network to the goal node is a series of rules and the confidence of each rule in the path is the likelihood of the consequent given the antecedent.

These advantages and the associated concepts will be explained and illustrated in the following subsections.

3.1. Proposed Approach for ARN Construction

We briefly describe our method for structuring association rules as a backward-directed hypergraph (hence referred to as B-graphs), pruning it to generate the ARN, and transforming it for reasoning. The method consists of four steps that will be expanded in subsequent sub-sections.

Step A. Given a database \( D \) and the minimum support and confidence, we first extract all association rules using a standard algorithm like Apriori [3].

Step B. Choose a frequent item \( g \) which appears as a singleton consequent in the rule set and represents the goal node, and build a leveled B-graph which recursively flows into the goal \( g \).

Step C. Prune the B-graph generated in Step B of hypercycles and reverse hyperedges. The resultant B-graph is called an ARN. We will give a formal justification for this step in the forthcoming sub-sections.

Step D. Find shortest paths between the goal node and the nodes at the maximal level (a variant of edge distance) in the ARN. The set of these paths represents the explanatory network for the goal node.

We next provide some basic background on directed hypergraphs that will aid us in defining an ARN and illustrating its various properties.

3.2. Directed Hypergraphs

A hypergraph is a pair \( H = (N, E) \) where \( N \) is a set of nodes \( \{n_1, n_2, \ldots, n_k\} \) and \( E \subset 2^N \) is the set of hyperedges. Thus each hyperedge \( e \) can potentially span more than two nodes. Contrast this with a directed graph where the edge set \( E \subset N \times N \subset 2^N \).

In a directed hypergraph the nodes spanned by a hyperedge \( e \) are partitioned into two parts, the head and the tail denoted by \( H(e) \) and \( T(e) \), respectively. A hyperedge \( e \) is called backward if \( |H(e)| = 1 \). Similarly an edge is called forward if \( |T(e)| = 1 \). A directed hypergraph is called \( B \)-directed hypergraph if all its hyperedges are backward. In the rest of the paper, we will refer to backward-directed hypergraphs as B-graphs. Thus, the set of association rules whose consequents are singletons map neatly into a B-graph. Each rule \( r \) is represented by a hyperedge \( e \), the antecedents of \( r \) by \( T(e) \) and the consequent by \( H(e) \).

We will also consider the antecedent of a rule as a single entity. For that we define the notion of a hypernode. Given a B-graph \( B \) with hyperedges \( \{e_1, \ldots, e_m\} \), the hypernodes induced by the hyperedge \( e_i \) are the tail \( T(e_i) \) and the head \( H(e_i) \) considered as a single entity. The set of all hypernodes is denoted by \( V \).

**EXAMPLE 1:** As can be seen in Fig. 3, \( N = \{A, B, C, D, E, F, G\} \), \( E = \{e_1, \ldots, e_6\} \), and \( V = \{A, B, C\} \).
\{D\}, \{C, D\}, \{E\}, \{E, F\}, \{G\}\). Thus \(F\) is a node but not a hypernode.

We now define a hyperpath and a hypercycle for a B-graph. A hyperpath is defined as a sequence \(P = \{v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_n\}\), where \(v_i, v_i\) is a hypernode, and \(e_i\) is a hyperedge. Furthermore for \(1 \leq i \leq n - 2\), \(v_i = T(e_i)\). \(H(e_i) \in v_{i+1}\). \(v_{n-1} = T(e_{n-1})\), and \(v_n = H(e_{n-1})\).

**EXAMPLE 2:** Again, as can be seen in Fig. 3, \(P = \{\{A\}, e_3, \{C, D\}, e_4, \{E, F\}, e_6, \{G\}\}\) is a hyperpath.

Finally, we define the size of a hyperpath \(|P|\) as the total number of hypernodes appearing on \(P\). Continuing with the previous example, \(|P| = 4\).

In the context of association rules, each node corresponds to a frequent item and frequent itemsets are mapped to undirected hyperedges [23,28,29]. There has been some theoretical work relating hypergraphs with association rules [28,29]. Here the relationship between frequent itemset discovery and the undirected hypergraph transversal problem has been noted in order to establish the computational complexity of frequent itemset mining. Directed hypergraphs [27,30] extend directed graphs and have been used to model many-to-one, one-to-many, and many-to-many relationships in theoretical computer science and operations research. Directed hypergraphs have also appeared with different names including “labeled graphs” and “And-Or” graphs. A set of association rules can be represented as a directed hypergraph. In [31], a set of rules have been modeled as directed hypergraphs in order to detect structural errors like circularity, unreachable goals, dead ends, redundancy, and contradiction.

### 3.3. Definition of ARN and Related Concepts

In this section, we will formally define an ARN, present an algorithm to generate it from a set of association rules and prove some important properties of the ARN.

Here, we define an ARN in terms of B-graphs.

**DEFINITION 1:** Given a set of association rules \(R\) and a frequent goal item \(z\) which appears as singleton in a consequence of a rule \(r \in R\). An ARN \((R, z)\) is a weighted B-graph such that

1. There is a hyperedge which corresponds to a rule \(r0\) whose consequent is the singleton \(z\).
2. Each hyperedge in \(ARN(R, z)\) corresponds to a rule in \(R\) whose consequent is a singleton. The weight on the hyperedge is the confidence of the rule.
3. The node representing \(z\) is reachable from any other node in \(ARN(R, z)\).
4. Any node \(p \neq z\) in the ARN is not reachable from \(z\).

This defines the basic structure of an ARN. Note that, given a database and a set of rules \(R\), the ARN for a given goal node \(z\) is unique, as shown in Theorem 2 in Section 3.4.1.

However, it is important to note that it is possible to have association rules of the form \(\{A, C\} \rightarrow B\) and \(B \rightarrow A\) in the rule set, from which an ARN is constructed according to the above definition. It is easy to see that the use of these two rules in the construction introduces a cyclic structure in the ARN. This cyclic structure is known as a hypercycle, and is more formally defined in Definition 2. Note that, in general, a hypercycle may consist of an arbitrary number of hyperedges, and can be substantially more complex than the two hyperedge (rule) example illustrated above.

**DEFINITION 2:** A hyperpath \(HC = \{v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_n\}\) is called a hypercycle if \(v_n \subseteq v_1\).

**EXAMPLE 3:** The hyperpath \(\{B, e_1, \{C, D\}, e_4, \{E, F\}, e_5, B\}\) is a hypercycle in the ARN shown in Fig. 3.

The occurrence of such hypercycles in a reasoning network such as an ARN is undesirable, since it prevents the reasoning algorithm to reach the goal node from any of the hypernodes involved in a hypercycle to the goal node. Hence, we removed hypercycles from an ARN using the concept of level defined below.

**DEFINITION 3:**
1. The level of the goal node is zero.
2. The level of a nongoal node \(v\), is defined as \(l(v) = \min\{l(u) + 1 | \exists v such that v \in T(e) and u = H(e)\}\).

Here, \(H(e)\) and \(T(e)\) denote the head and tail of the hyperedge \(e\), respectively.

3. The level of a hypernode \(c\) is defined as \(l(c) = \min\{l(s) | s \in c\}\).

**EXAMPLE 4:** For the ARN in Fig. 3, \(l(G) = 0, l(F) = l(E) = 1, l(C) = l(D) = 2,\) and \(l(B) = l(A) = 3\). For hypernodes, \(l([C, D]) = 2\) and \(l([E, F]) = 1\).

The justification for the removal of hypercycles is discussed further in detail in Section 3.5. This step is also supported by similar requirements of other graphical modeling frameworks, such as Bayesian networks [32], which do not allow cycles in the underlying graph.

Definition 1 of ARNs also does not prevent another reasoning anomaly introduced by the bi-directional nature of association rules. Consider the hyperedge \(e_2\) in the ARN shown in Fig. 4 (Section 3.5), for which the goal node is \(A\). Although this hyperedge is not part of a hypercycle,
Algorithm 1 (shown below) takes as input the rule set \( R \) and a goal node \( c \) which appears as singleton consequent in \( R \). The \( \text{Rules.getRules}(R, u, s) \) is a function that returns all rules in \( R \) whose consequent is \( u \), but antecedent does not contain \( s \). For each of these rules, \( \text{Rules.getAntecedents}(u) \) returns the set of all antecedents. The level of each of these antecedent elements is determined on the basis of Condition 1 and Definition 4. For all the rules satisfying Condition 1, a hyperedge is added to the ARN. Finally, hypercycles between hypernodes which are on the same level are removed using \( \text{Rules.removeLevelCycles}() \). The algorithm returns the generated ARN.

3.4. Results about ARN construction algorithm

In this section we prove several properties of ARN generation. In Theorem 1, we prove the time complexity of constructing an ARN. Theorem 2 shows that for a given set of association rules and goal node, the ARN is uniquely determined. Theorem 3 proves that the goal node is reachable from all other nodes in the ARN. Finally, Theorem 4 shows that the ARN does not contain any hypercycles.

**THEOREM 1:** The time complexity of ARN generation is \( O(nk) \), where \( n \) is the number of frequent items and \( k \) is the number of rules whose consequents are singletons.

**PROOF:** For each node in the ARN, the hyperedges flowing into it are found by searching the set of all rules whose consequents are singletons. Furthermore, the number of nodes appearing in the ARN is bounded by the number of frequent items. Hence the complexity is \( O(nk) \). Also \( k \) is bounded by \( \sum_{i=2}^{l} n_i^l \) where \( l \) is the length of the longest frequent item set and \( n_i \) is the number of frequent itemsets of size \( i \).

**THEOREM 2:** The ARN generated for a given goal node and a given rule set is unique, i.e., it does not depend upon the sequence in which the nodes are explored.

**PROOF:** Let \( u \) and \( v \) be two nodes at the same level. We want to show that by exploring \( u \) and \( v \) in different orders we are not losing a hyperedge whose consequent is one of them and the antecedent is the other—this is the only choice that is made in the algorithm which could result in different ARNs. WLOG\(^4\) assume \( u \) is explored before \( v \). Let \( \{e_1, \ldots, e_m\} \) be the set of hyperedges for which \( u \) is a consequent and \( v \) is one of the antecedents. Then, by the definition of level, \( \text{level}(e_i) = \text{level}(v) \) for all \( 1 \leq i \leq m \). Note that, by Condition 1, there cannot be

\(^4\) WLOG = without loss of generality.
Algorithm 1: Algorithm for constructing an ARN from a rule set R and flowing into consequent c using a breadth-first strategy.
any hyperedge whose consequent is \( u \) and any of whose antecedents have level less than \( u \) (which is the same as the level of \( v \)).

Now, once we explore \( v \), all the hyperedges, whose consequent is \( v \) and whose antecedents contain \( u \), are free to flow into \( v \) because of Condition 1. Thus we have not lost any hyperedges between \( u \) and \( v \) since they are at the same level. A similar argument holds when \( v \) is explored before \( u \). Thus the set of hyperedges and hypernodes remains the same in both cases. Hence, the same ARN will be generated for a given goal node, irrespective of the order in which the nodes at the same level are explored.

**THEOREM 3:** The goal node is reachable from any node in the ARN.

**PROOF:** By induction on level. By definition, the level of the goal node is zero and trivially reachable from itself. Assume that the goal node is reachable from any node at level \( i \). For each node \( u \) at level \( i+1 \), there exists at least one hyperedge \( e \) such that \( l(H(e)) = i \) and \( u \in T(e) \). Thus the result holds for all \( i+1 \) and hence, for all levels \( i < \max l(u) | u \in N | \).

**THEOREM 4:** The ARN generated by the algorithm is free of hypercycles across levels.

**PROOF:** By contradiction. Let \( C \) be a hypercycle across levels, i.e., \( C = \{v_0, e_1, \ldots, e_n, v_0\} \) where \( v_0 \subset v_0 \). Let \( v_i \) be the first hypernode in \( C \) where \( l(v_i) < l(v_0) \). Then, by construction of an ARN, \( l(v_{i-1}) < l(v_i) \). Therefore, \( l(v_{i+1}) < l(v_i) \). Now \( l(v_{i+1}) \geq l(v_i) \) because \( v_{i+1} \) is a singleton and \( v_{i+1} \subset v_i \). Therefore, \( e_{i+1} \) is a hyperedge from an antecedent at a lower level to a consequent at a higher level. This contradicts Condition 1.

### 3.5. Removal of Hypercycles and Reverse Hyperedges, and their Relation to Rule Pruning

In Section 3.3, we illustrated how hypercycles and reverse hyperedges introduce redundancy and cyclicity into an ARN constructed using its basic definition. In this subsection, we provide a mathematical justification for the removal of hypercycles and reverse hyperedges. We will show that in either case the information provided by the pruned hyperedge, or the interestingness of the corresponding rule, is small, given the rest of the ARN.

Consider the example in Fig. 4. Given the goal node \( A \), during ARN construction, the two hyperedges \( e_2 \) and \( e_7 \) are removed. The removal of hyperedge \( e_7 \) is more “crucial” than \( e_2 \) because it is part of a hypercycle; otherwise, \( A \) will never be reachable from \( F \), violating the definition of the ARN. On the other hand, the removal of the reverse hyperedge \( e_2 \) is justified because it is part of a redundant path from \( B \) to \( A \). Hence, the removal of \( e_2 \) will not destroy the reachability condition of the ARN. This illustrates the difference between the two kinds of pruning operations. We provide separate formal arguments for each of the two operations below.

#### 3.5.1. Removal of Hypercycles

We first note that a hypercycle \( H \) in a B-graph cannot be of size less than three. We will provide the desired justification for two cases, one in which the size of \( H \) is three and the other in which the size of \( H \) is four. From transitivity, it follows that the removal of any hypercycle of size bigger than four can be justified using the arguments presented for the second case. We will now consider the two cases.

**CASE 1:** Consider two rules of the form \( A \rightarrow B \) and \( B \rightarrow A \). If \( A \) and \( B \) are the same level, then WLOG remove the hyperedge with lower confidence to break the hypercycle. Else, WLOG, assume level \( (A) > \text{level}(B) \). Since we are exclusively dealing with rules whose antecedents are a conjunction of items and the consequent is a singleton item, for a vast majority of these rules, \( P(A) < P(B) \). In such a scenario,

\[
\text{conf}(A \rightarrow B) > \text{conf}(B \rightarrow A).
\]

This justifies the removal of the hyperedge representing \( B \rightarrow A \).

**CASE 2:** Consider three rules of the form \( A \rightarrow B, B \rightarrow C \), and \( C \rightarrow A \). Our argument is based on the concept of information gain. Assume we have an information channel 1 which is a sequence of transactions. Then, given that the pair of itemsets \( (A, B) \) and \( (B, C) \) are frequent, i.e., they appear close to each other with high probability then the information that \( (A, C) \) is frequent is not surprising. We formalize this argument as follows.

**DEFINITION 5:** Let \( F \) be the set of all itemsets. Let \( d : F \times F \rightarrow [0, 1] \) be defined as

\[
d(A, B) = 1 - \frac{P(A, B)}{P(A) + P(B) - P(A, B)},
\]

where \( A, B \in F \) and \( P(A) \) is the support of the itemset \( A \).

**THEOREM 5:** The function \( d \) is a metric on the space \( F \).

**PROOF:** The function \( d \) is identical to the distance measure \( \rho \) defined in [25]. This proof follows directly from the proof of Lemma 3.2 in [25].
COROLLARY 1: For $o < \delta \ll 1$, the information gain from observation that the pairs (A,C) are close to each other is small given that $d(A, B) < \delta$ and $d(B, C) < \delta$.

PROOF: Follows directly from the triangle inequality of $d$.

Now, WLOG, assume that level($A$) > level($C$). The fact that $d(A, C)$ is small given that $d(A, B)$ and $d(B, C)$ are small is an indication of the fact that the rule $A \rightarrow C$ and $C \rightarrow A$ can be derived from the rules $A \rightarrow B$ and $B \rightarrow C$ which are already in the ARN. Thus, $C \rightarrow A$ can be safely pruned, without violating the reachability constraint of the ARN.

3.5.2. Removal of reverse hyperedges

The removal of reverse hyperedges in an ARN can be justified on the basis of the fact that they generate redundant paths from a node to the goal. This can be formally proved as follows.

THEOREM 6: Let $a$ be a node in the ARN from which a reverse hyperedge $e$ originates. Then, there exists a path $P$ from $a$ to the goal node such that $e \in P$ and the size of $P$ is smaller than the size of the path from $a$ to the goal node in which $e$ participates.

PROOF: Let $l(a) = l_1$ and $l(H(e)) = l_2$. Since $e$ is a reverse hyperedge, $l_1 < l_2$. We observe that the path of smallest size from any node at level $l$ in the ARN to the goal node is of size $l + 1$ (follows from the definition of level). Let this path of smallest size from $a$ be $P_1$ and the one from $H(e)$ be $P_2$. Clearly, the size of $P_1$ is $l_1 + 1$ and that of the new path $b; e \cup P_2$ is $l_2 + 2$. Thus $P_1$ is the required path.

This theorem justifies the redundancy of the rule represented by a reverse hyperedge and hence its removal.

3.6. Benefits of ARN

In the introduction we raised the question of the utility of association rules beyond simple exploratory analysis. ARNs are also a tool for exploratory analysis that provides a context for understanding and relating the discovered rules with each other. More specifically, ARNs offer the following benefits.

Organizing rules in a context: The primary utility of ARN is in the organization of a potentially large set of association rules, such that a reasoning goal may be explained using the most relevant rules in the set. This explanation can extend beyond the immediate rules that have the goal as a consequent, and thus provides a framework for organizing these rules in a context. The case studies presented later provide evidence of the utility of ARNs for generating statistically viable hypotheses from a set of association rules derived from raw data.

Local pruning: ARNs provide a graphical method to prune rules by associating redundant rules with hypercycles and reverse hyperedges. Furthermore, the pruning takes place in the context of a goal node. Thus, a rule that is redundant for a particular goal node may become relevant for another goal. This is more flexible than pruning based on statistical measures of interestingness.

Consider the ARNs shown in Figs. 5 and 6. When the goal node is $G$ (Fig. 5), the hyperedge $EB$ is a redundant rule, which may be eliminated. On the other hand, when the goal node is $D$ (Fig. 6), the same hyperedge $EB$ is relevant. Thus, pruning of a rule as per our notion becomes dependent upon the context of the goal node. We refer to this kind of pruning as local pruning.

Reasoning using path traversal: An ARN is a weighted hypercycle free B-graph. Hyperpaths that lead to the goal node can be interpreted as providing an explanation for the goal node.

Formally, let $V_{max}$ be the set of all maximum level nodes in the ARN. For each $v \in V_{max}$, let $P_v$ be the set of all hyperpaths from $v$ to the goal node $g$. We can then define two cost measures on each hyperpath $p \in P_v$:

\[
\text{Weight}(p) = \sum_{e_i \in p} \log([\text{conf}(e_i)])
\]

\[
\text{Info}(p) = -\sum_{e_i \in p} \text{conf}(e_i) \log([\text{conf}(e_i)])
\]

Fig. 5 ARN with goal node $G$. The edge $e_5$ is not part of the ARN because it participates in a hypercycle.

Fig. 6 ARN with goal node $D$. The edge $e_5$ is now part of the ARN. This illustrates the adaptive nature of local pruning.

Statistical Analysis and Data Mining DOI:10.1002/sam
where \( e_i \) is a hyperedge and \( \text{conf}(e_i) \) is the confidence of the rule represented by \( e_i \).

The reason for introducing two cost functions is that they provide different kinds of information depending upon the context. For example, \( \text{Weight}(p) \) can be interpreted as the strength of the correlation between the source and the goal node. Similarly, \( \text{Info}(p) \) can be interpreted as the total information gain along the path from the source to the goal node.

Now, the optimal path in \( P_e \) using \( \text{Weight}(p) \) or \( \text{Info}(p) \) is likely to be the best explanation for the dependence of the goal node on \( v \). Computing these optimal hyperpaths for all \( v \in V_{\text{max}} \) provides a reasoning network for the goal node \( g \). The problem of optimal hyperpaths in B-graphs has been studied by [33] where they have reported an algorithm of time complexity \( O(|H| + n \log n) \) where \( |H| \), the size of the hypergraph, is \( \sum_{v \in E} (|T(e_i)| + 1) \).

4. CASE STUDIES

To recall, our overall objective in this paper is to provide a systematic methodology of combining data mining techniques with traditional statistical methods to speed up the process of “theory building” and validation. The role of data mining in this methodology is to search the “space of theories” and offer credible candidates and the role of statistical methods is to verify the theories offered by data mining. To demonstrate the utility of this methodology, we provide two extensive case studies using the techniques described in the previous section. The first case study is on the Open Source Software (OSS) development data and the other study is on the Business Longitudinal Survey (BLS) data provided by the Australian Bureau of Statistics (ABS).

4.1. Case Study 1—OSS development

OSS represents communities of software developers who use the Internet infrastructure to coordinate the work of building robust and scalable software. The objective of the case study is to determine the factors which contribute to the success (measured in number of downloads) of OSS projects. Since research in OSS is relatively new, it is not clear what factors contribute to the success of such projects. This makes it an ideal case study in our situation, as data mining may offer new insights (in terms of attribute–value pairs) which may explain the success or failure of OSS projects. As we will show, we can use ARNs to organize the output of data mining and then hypothesize a model for OSS which can be tested by standard statistical methods. Extensive data on thousands of such OSS projects is available from the SourceForge website. In this subsection we describe some of the main results.\(^5\) However, we briefly digress to provide a summary of the data extraction, data cleaning, and data engineering phases that were carried out [34].

**Data extraction:** We downloaded OSS data from the SourceForge website (www.sourceforge.net) in 2003. At the time of the study, the SourceForge website had 40 002 projects hosted (including SourceForge) and 424 862 registered users. Only projects which had a relatively homogeneous profile were investigated. The dataset was obtained by a program that crawls through the project summary page for each of these projects and stores the necessary information in a text file. The dataset holds features of each project and their activity information. Table 1 describes the facts about the dataset:

<table>
<thead>
<tr>
<th>Data description</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number or records</td>
<td>24 002</td>
<td></td>
</tr>
<tr>
<td>Total number of variables</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>Number of categorical variables</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Number of numeric variables</td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>

**Data cleaning:** After the data were extracted, we noticed that many entries related to several attributes were mislabeled. For example, the attribute audience which captured the end-user for the application was populated by the entries for the environment variable. Similarly, the time attribute was either missing or lacked a consistent format. We performed appropriate cleaning operations to reduce such errors.

**Data engineering:** An important transformation we carried out was to convert the numeric attributes to ordinal ones using equal-sized binning. The number of bins used was up to four representing missing, low, medium, and high values. This was necessary since most off-the-shelf association rule programs accept only categorical data. However, as we will show, this did not result in a substantial loss of information as similar results were obtained using factor analysis which operated on the original numerical values.

After carrying out a detailed preliminary analysis of the data, we decided to focus on 12 attributes which together indicate project activity along different dimensions. The attributes were discretized to discover association rules but the original numerical values were retained for factor analysis. These attributes are listed in Table 2.

4.1.1. Constructing ARN

We constructed the ARN shown in Fig. 7 using Algorithm 1 described earlier: Given a set of rules \( R \) and an item (attribute–value pair) \( z \), we find all rules in \( R \) in which \( z \) appears as a consequent. We recursively extract more rules, where the antecedents in the previous step now

---

\(^5\) Detailed results of this case study can be found in [7].
Table 2. The attributes used to construct the ARN.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Attribute description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrators</td>
<td>Number of administrators</td>
</tr>
<tr>
<td>Developers</td>
<td>Number of developers</td>
</tr>
<tr>
<td>CVS commits</td>
<td>Number of CVS commits</td>
</tr>
<tr>
<td>Bugs found</td>
<td>Number of bugs found</td>
</tr>
<tr>
<td>Bugs fixed</td>
<td>The percentage of bugs fixed</td>
</tr>
<tr>
<td>Support requests</td>
<td>Number of support requests</td>
</tr>
<tr>
<td>Support requests completed</td>
<td>Percentage of support requests completed</td>
</tr>
<tr>
<td>Patches started</td>
<td>Number of patches started completed</td>
</tr>
<tr>
<td>Patches completed</td>
<td>Number of patches completed</td>
</tr>
<tr>
<td>Public forums</td>
<td>Number of public forums</td>
</tr>
<tr>
<td>Mailing lists</td>
<td>Number of mailing lists</td>
</tr>
<tr>
<td>Forum messages</td>
<td>Number of forum messages</td>
</tr>
</tbody>
</table>

play the role of consequents. A support level of 6% and a confidence level of 50% was used to construct the ARN.

Analysis: Figure 7 shows the ARN extracted from the OSS data for the goal of high downloads. The ARN clearly shows that association rules naturally discover attributes that work in concert. For example, the three variables; high activity in public forums, forum messages, and mailing list encapsulate communication activity related to the project and have a first-order effect on the success of the software project. This validates the importance of effective communication which has often been cited as a predictor for success in software engineering projects [35].

4.1.2. Clustering and factor analysis

An ARN provides information about the hierarchical relationships between the variables (attributes) of the system under investigation. A natural follow-up is to cluster the items that are elements of the ARN. Hypergraph clustering based on min-cut hypergraph partitioning is a well-researched topic and has been used, among other things, for VLSI design [36]. Han et al. [23] have used it for itemset clustering.

One issue that arises when the clustering problem is being set up is to decide on which similarity measure to use. For example, in [23], the weighted confidence rule is used: if a hyperedge spans the items \{A, B, C\}, then the weight of the hyperedge is the average of the confidence of the rules \{A, B\} → \{C\}, \{A, C\} → \{B\}, and \{B, C\} → \{A\}. Another similarity measure that can be used is related to the \textit{lift} of an itemset [35]. Again consider three items \{A, B, C\}. Then the \textit{lift} of \{A, B, C\} is \frac{\sigma(A, B, C)}{\sigma(A) \sigma(B) \sigma(C)}.

We have used weighted confidence in our work because it is more directly related to rules rather than itemsets.

![Fig. 7 ARN constructed using Algorithm 1. The ARN is a directed hypergraph. The distinguished goal node is Downloads.](image-url)
Another purpose of the case study is to generalize the results generated from the ARN. We partitioned the ARN into four components using min-cut hypergraph partitioning. The results of clustering are shown in Fig. 8. The items naturally fall in clusters which we have labeled communication, support, development activity, and organization/commitment. In order to check the validity of our method we have used a completely different approach, namely factor analysis, on the original nondiscretized dataset and compared the two approaches.

Analysis: Factor Analysis is a statistical technique that attempts to find latent cluster(s) of variables which best describe the data [37]. It is based on principal component analysis (PCA). It operates on the correlation matrix of the variables as opposed to the covariance matrix. The 12 variables in Table 1 are represented by four factors 1, 2, 3, and 4. Each of these factors captures the essence of the 12 original variables. Table 3 shows how the result of the ARN clusters and factor analysis corresponded. For example, factor 1 captures the effect of variable Bugs Found, Patches Started, and Patches Completed which relates the cluster labeled Development Activity. Notice that only two variables, namely, Concurrent Versions System (CVS) Commits and Forum Messages fall in nonmatching factors and clusters.

There is a remarkable degree of congruence between the clusters arrived at via ARNs and Factor Analysis, and lend weight to the whole concept of inducing ARNs from data. Also notice that the variables that become part of clusters derived from the ARN are semantically related. For example, the cluster Organization and Commitment captures variables # of Administrator and # of Developers and their respective factor loadings within factor 4 are very high (0.756 and 0.800, respectively).

4.2. Case Study 2—Organizational Practices and Firm-level Productivity

This case study focuses on productivity effects of a large number of organizational practices. We use a large firm-level dataset to analyze the complex relationships between a variety of organizational practices and their complementary impact on firm-level, long-term multifactor productivity (MFP) growth. Similar to the previous study, these complex relationships motivate the use of data mining
Table 3. Relationship between the ARN clusters and the factors derived from Factor Analysis.

<table>
<thead>
<tr>
<th>Factor 1</th>
<th>Development activity</th>
<th>Support</th>
<th>Communication intensity</th>
<th>Organization and commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Bugs found (0.791)</td>
<td>#Patches started (0.967)</td>
<td>#Patches completed (0.968)</td>
<td>#Forum messages (0.711)</td>
<td></td>
</tr>
<tr>
<td>Factor 2</td>
<td>#Support requests (0.686)</td>
<td>%Support requests completed (0.521)</td>
<td>#Public forums (0.903)</td>
<td></td>
</tr>
<tr>
<td>Factor 3</td>
<td>#CVS commits (0.557)</td>
<td>#Public forums (0.903)</td>
<td>%Support requests completed (0.521)</td>
<td>#Administrators (0.756)</td>
</tr>
<tr>
<td>Factor 4</td>
<td>#CVS commits (0.557)</td>
<td>#Mailing list (0.561)</td>
<td>%Support requests completed (0.521)</td>
<td>#Developers (0.800)</td>
</tr>
</tbody>
</table>

The values in the parenthesis are factor loadings.

Table 4. Descriptions of capital investments, labor and output.

<table>
<thead>
<tr>
<th>Description</th>
<th>Availability</th>
<th>Summary statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capital (K)</td>
<td>3 years</td>
<td>$819K (1995–1996)</td>
</tr>
<tr>
<td>Labor (L)</td>
<td>4 years</td>
<td>27.36 (1994–1995)</td>
</tr>
<tr>
<td>Value-added (Q)</td>
<td>4 years</td>
<td>$3184K (1994–1995)</td>
</tr>
</tbody>
</table>

6 Detailed results of this case study can be found in [38].

in general, and ARNs in particular, for addressing this problem. In this subsection we describe some of the main results of using ARNs for this study. The main dataset used in this study is the ABS BLS covering four consecutive years: 1994–1995, 1995–1996, 1996–1997, and 1997–1998. The longitudinal survey was designed to provide information on the growth and performance of Australian businesses, and to identify various economic and structural characteristics of these businesses [39]. In each year, the survey was designed to target variables based on a particular organizational theme. In 1994/1995 the focus was on benchmarking, in 1995/1996 the focus was on various organizational processes, in 1996/1997 the focus was on production technology and computer use, and in 1997/1998 the focus was on training. The BLS contains about 9 550 confidentialized respondent records and 3864 records that participated in all 4 years. This BLS survey contains a total of 787 variables with a wide coverage of organizational characteristics and measures including various business intentions, capital expenditures, organizational practices, and performance variables. Descriptions of the capital, labor, and output variables are provided in Table 4.

Data cleaning: A total of 32 organizational variables in the dataset were found useful for our study, and extracted from the dataset for ARN construction. Data cleaning was not a major issue for this dataset because ABS followed a well-established protocol to ensure data items collected from survey instruments have high validity and reliability. In our present case, to ensure each record has value for each organizational variable, we select only firms that participated in all four years. Out of the 3864 panel data records, 565 records were discarded due to incomplete data values. The remaining 3299 records that have values for all 32 organizational variables are used for this analysis.

Data engineering: We calculate the MFP growth for each firm using an established approach called the Growth Accounting Framework [40]. It was found that the average MFP growth of all 3299 business units is −6.9%, in which...
Table 5. Firm-level MFP growth results.

<table>
<thead>
<tr>
<th>Number of business units</th>
<th>Average MFP growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive MFP growth</td>
<td>1579 (48%)</td>
</tr>
<tr>
<td>Negative MFP growth</td>
<td>1720 (52%)</td>
</tr>
<tr>
<td>Average MFP growth</td>
<td>3299 (100%)</td>
</tr>
</tbody>
</table>

1579 (48%) of 3299 business units obtained positive MFP growth and 1720 (52%) business units obtained zero or negative MFP growth. The details of the MFP results are provided in Table 5.

Furthermore, the preliminary observations of organizational data reveal that the variables capturing the organizational practices in the dataset were mostly discrete variables except StockAdjust, which was a continuous variable. Two organizational variables were modified to provide a workable solution for the analysis. These are StockAdjust and RD (refer to Table 5). First, the variable StockAdjust was created to capture the changes (in percentage) of each firm’s inventory level with a 4-year difference (between 1994/1995 and 1997/1998) instead of 1-year difference. These values were subsequently discretized into three categories (i.e. increase, decreased, and not applicable (NA)). Second, because firms reported whether they performed research and development (R&D) in each year’s BLS survey, the variable RD was created to generalize all four years’ values into one variable. This RD variable was to show if firm performed R&D in any one of the 4 years. All 32 variables listed in Table 5 are either binary coded [Yes/No] or categorical with three values [increase/no increase/NA] or [>25% ≤25% or NA]. The list of organizational practices and their corresponding variables is provided in Table 6.

4.2.1. Constructing ARN

At this stage, the ARN technique was applied to the 32 organizational variables and the implied MFP derived from the growth accounting framework. In this case, first the goal was fixed (for instance, either positive or negative MFP growth), and all the significant rules that satisfied the minimum thresholds were generated. Then, the antecedents (namely the first-order variables leading to the goal) of these rules were considered as consequents in the following round of association rule generation and all the rules of these antecedents as consequents were generated. There were modifications in this analysis as compared with the previous case study. Instead of having a common minimum support and minimum confidence thresholds for the ARNs, the two ARNs were allowed to have different minimum support and minimum confidence thresholds. Also, the minimum support and minimum confidence thresholds were allowed

Table 6. Organizational practices employed for association analysis.

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Organizational practices</th>
<th>Variable name</th>
<th>Organizational practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompPrice</td>
<td>Comparing prices</td>
<td>BusNetwork</td>
<td>Implementation of business network</td>
</tr>
<tr>
<td>CompCosts</td>
<td>Comparing costs</td>
<td>BusComp</td>
<td>Business comparison</td>
</tr>
<tr>
<td>CompQual</td>
<td>Comparing quality of products</td>
<td>ExpMarketing</td>
<td>Export marketing</td>
</tr>
<tr>
<td></td>
<td>or service</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CompMkt</td>
<td>Comparing marketing or</td>
<td>BusLink</td>
<td>Implementation of business link</td>
</tr>
<tr>
<td></td>
<td>advertising strategies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CompCServ</td>
<td>Comparing client services</td>
<td>ProdTech</td>
<td>Using production technology</td>
</tr>
<tr>
<td>CompProdR</td>
<td>Comparing range of products</td>
<td>UC</td>
<td>Use of computer</td>
</tr>
<tr>
<td></td>
<td>and service</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BPI</td>
<td>Business process improvement</td>
<td>StructuredTraining</td>
<td>Providing structured training</td>
</tr>
<tr>
<td>Adjust_PL</td>
<td>Significantly increase</td>
<td>JobTraining</td>
<td>Providing on-the-job training</td>
</tr>
<tr>
<td></td>
<td>production levels</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IntroGood</td>
<td>Introduction of new products</td>
<td>Workshops</td>
<td>Providing seminars, workshop and conferences etc.</td>
</tr>
<tr>
<td></td>
<td>or services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TQM</td>
<td>Total quality management</td>
<td>JobRotation</td>
<td>Providing job rotation and exchanges</td>
</tr>
<tr>
<td>QA</td>
<td>Quality assurance</td>
<td>MgTraining</td>
<td>Providing management training</td>
</tr>
<tr>
<td>JIT</td>
<td>Just-in-time management</td>
<td>ProTraining</td>
<td>Providing professional training</td>
</tr>
<tr>
<td>StraPlan</td>
<td>Implementation of strategic</td>
<td>ITRaining</td>
<td>Providing training for computer specialist</td>
</tr>
<tr>
<td></td>
<td>plan</td>
<td>SupplierTraining</td>
<td>Providing trade and apprenticeship training and traineeships</td>
</tr>
<tr>
<td>BusPlan</td>
<td>Implementation of business plan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IE_Report</td>
<td>Income and expenditure report</td>
<td>RD</td>
<td>Perform research and development in any of the four years</td>
</tr>
</tbody>
</table>
to be dynamically adjusted for first- and second-order association rules generation.

The main advantages of these modifications allowed the focus to be on the most important set of rules at each level (i.e. top K rules with a highest possible minimum support and minimum confidence thresholds) rather than having arbitrary minimum support and minimum confidence thresholds that are common for the generation of the two ARNs. This is because problems can arise when (i) one of the two ARNs might require lower thresholds for any association rule to appear, and thus, by setting the thresholds too high for one ARN would cause the weaker ARN to disappear, and (ii) if any of the ARNs requires lower thresholds to generate higher-order association rules than to generate lower-order association rules, setting fixed thresholds would limit the chance of discovering higher-order rules. The challenge is that the minimum support and minimum confidence thresholds needed to be set low enough to generate at least one first-order association rule. However, setting the global minimum support and minimum confidence set for the second-order rules generation to be different (if necessary) from the minimum support and minimum confidence set for the first-order rules generation.

Analysis: For each ARN, we began with very high minimum support and confidence thresholds. We then reduced the support and confidence thresholds slowly until the first association rule was found. The first association rule would provide a set of most frequent variables leading to MFP growth. These variables are the first-order variables. To discover the second-order variables leading to first-order variables, we maintained the same support level but allowed the confidence level to vary. If too many rules were found, we increased the confidence threshold, and reduced the confidence threshold otherwise. This way, we would get the most frequent variables leading to first-order variables. For the ARN with positive MFP growth as the goal, the minimum support was set as 10% and minimum confidence as 55% for the first-order rules, and the minimum confidence was increased to 99% for the second-order rules with minimum support maintained at 10%. For the ARN with negative MFP growth as the goal, the minimum support was set as 15% and minimum confidence as 65% for the first-order rules, and the minimum confidence was increased to 90% for the second-order rules with the minimum support maintained at 10%. Figure 9 shows the ARN with positive MFP growth.

Applying the same strategy to this ARN with the goal set to negative MFP growth, the ARN construction was terminated after the second-order association rules were generated, since no new third-order variables were found leading to the second-order variables with the given support. Figure 10 shows the ARN with negative MFP growth.

4.2.2. Statistical analysis and model development

Several interesting patterns emerge when both ARNs are analyzed with reference to each other. First, the two ARNs generated are quite different. This result suggests that factors that contribute to positive MFP growth are not
necessarily the same as those contributing to negative MFP growth. For example, the ARN in Fig. 9 shows that the three variable instances (i.e., reduced stock-level, use of computers, and use of budget plan) have the first-order effect on positive MFP growth, whereas the other ARN in Fig. 10 shows a different set of variable instances [i.e., increase stock-level, no implementation of Total Quality Management (TQM) and no introduction of goods] has the first-order effect on negative MFP growth. Second, the positions of some factors in the two ARNs are different. These results also suggest that some factors affect productivity growth in the positive direction, but the absence of these factors may not have the level of reverse effects. Third, the relationship structure among organizational practices that leads to productivity growth are not necessarily the same relationship structure set of organizational practices that hinder productivity growth. After examining all the variables in both the ARNs, we classify the variables in the two ARNs using the following criteria:

- Variables with full ellipse if they exist in both ARNs.
- Variables with dotted ellipse if they exist in only one of the ARNs.
- First-order variables are labeled with a double line.
- Second-order variables are labeled with a single line.

The objective of this step is to convert variable instances from data-mining results to candidate-type variables (i.e., to develop a testable model based on the two ARNs). The productivity impacts of these candidate-type variables can be hypothesized and tested statistically. On the basis of the classification of the variables, there are only nine variable instances common to both ARNs. These nine variables are shown in Table 7.

| Table 7. Variable instances have bi-directional impact on MFP growth. |
|--------------------------------------------------|--------------------------------------------------|
| Variable                      | Figure 9: ARN (positive MFP growth) | Figure 10: ARN (negative MFP growth) |
| Change in stock level         | First order | First order |
| PC use                        | First order | Second order |
| Budget plan                   | First order | Second order |
| Business plan                 | Second order | Second order |
| Quality assurance             | Second order | Second order |
| Income/expenditure report     | Second order | Second order |
| Adjust production level       | Second order | Second order |
| R&D                           | Second order | Second order |
| Business comparison           | Second order | Second order |

In order to synthesize the data-mining results, the number of candidate variables to be considered was reduced by selecting only the first-order variable instances from either ARN. The five first-order candidate variables were classified as strong and weak candidate variables. To be classified as a strong candidate variable, two necessary conditions are required. First, variable instances have to exist in both ARNs with opposite directions. Second, these variable

Fig. 10 ARN with negative MFP growth.
instances have first-order impact on either positive or negative MFP growth. However, variables that have first-order impacts on only either positive or negative MFP growth (i.e., they only exist in one ARN) are classified as weak candidate variables. Table 8 shows the first-order variable instances discovered in the ARNs.

There are total of five different first-order candidate variables. Of these variables, three variables (namely Adjustment in stock level, Use of computer, and Budget Plan) appeared in both ARNs with opposite directions. The other two variables (namely Introduction of goods and Implementation of TQM) appeared only in one ARN. One positive note is that among the other variables, use of computers is shown to be a strong candidate variable that has strong influence on MFP growth.

The statistical findings have revealed interesting insights about how these variables behave in organizations because variables which appeared in both ARNs are found to have higher statistical significance than variables which only appear in just one of the ARNs. This phenomenon points to two interesting research issues: (i) it has been confirmed that certain factors that affect organizational performance positively do not necessarily have a reverse effect when they are reduced or removed, and (ii) statistical inferences are perhaps inadequate to estimate the impact of such variables that affect the dependent variable in one direction only. In the next step, relationships among the organizational variables were modeled and statistically analyzed.

Based on the regression results, we construct our preliminary model using the relationship structure discovered in the two ARNs, i.e., by careful interpretation of the relationship structure from Figs. 9 and 10. First, the first-order variables that found to be statistically significant at 0.01 levels were stock level adjustment (StockAdjust), Computer use (UC), implemented Budget Plan (BudgetPlan), and implemented TQM (TQM). The variable representing introduction of goods and services (IntroGood) is removed due to its statistical insignificance from regression results.

On the basis of the relationship structure revealed in Fig. 9, the control variable is also modeled in the regression to adjust for the possible industry effect.

\[
\delta_3 = \lambda + \beta_1 UC + \beta_2 StockAdjust + \beta_3 BudgetPlan + \beta_4 TQM + \beta_5 IntroGood + \beta_6 Industry + \epsilon.
\]

By estimating the significance of each coefficient (\(\beta_i\)), one can statistically verify the relationship between the dependent variable (MFP growth) and the explanatory variables. The regression results are provided in Table 9.

---

### Table 8. Candidate variables for statistical testing.

<table>
<thead>
<tr>
<th>Positive MFP Growth</th>
<th>Negative MFP growth</th>
<th>Candidate variable</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decrease stock-level (first order)</td>
<td>Increase stock level (first order)</td>
<td>Strong negative</td>
<td>StockAdjust</td>
</tr>
<tr>
<td>Use computer (first order)</td>
<td>No use computer (second order)</td>
<td>Strong positive</td>
<td>UC</td>
</tr>
<tr>
<td>Use budget plan (first order)</td>
<td>No use budget plan (second order)</td>
<td>Strong positive</td>
<td>BudgetPlan</td>
</tr>
<tr>
<td>No introduction of goods (first order)</td>
<td></td>
<td>Weak positive</td>
<td>IntroGood</td>
</tr>
<tr>
<td>No implementation of TQM (first order)</td>
<td></td>
<td>Weak positive</td>
<td>TQM</td>
</tr>
</tbody>
</table>

### Table 9. Coefficients estimates of main effect model.

<table>
<thead>
<tr>
<th>Estimates</th>
<th>Standard errors</th>
<th>t-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC</td>
<td>0.229***</td>
<td>0.022</td>
</tr>
<tr>
<td>StockAdjust</td>
<td>-0.082***</td>
<td>0.010</td>
</tr>
<tr>
<td>BudgetPlan</td>
<td>0.063***</td>
<td>0.012</td>
</tr>
<tr>
<td>TQM</td>
<td>0.047***</td>
<td>0.011</td>
</tr>
<tr>
<td>IntroGood</td>
<td>0.016</td>
<td>0.011</td>
</tr>
<tr>
<td>Control</td>
<td>Industry</td>
<td>R^2</td>
</tr>
<tr>
<td>N</td>
<td>660</td>
<td>F-value</td>
</tr>
</tbody>
</table>

*** p < 0.01; ** p < 0.05; * p < 0.10.
moderating relationships are between computer use and stock level adjustment, and between computer use and budget plan. On the basis of relationship structure revealed in Fig. 10, the two mediating relationships are between computer use and budget Plan, and MFP growth, mediated by TQM. The candidate relationships are provided in Table 10.

We consider these candidate analytic properties from Table 10 to construct the preliminary path model for further statistical testing. Figure 11 illustrates the proposed path model.

In summary, these two case studies illustrate how ARNs can be used effectively for identifying complex relationships between factors that lead to the achievement of a certain goal. These effects can be both immediate and distant, and the structure of an ARN provides indications about these dependencies. These facts justify the use of ARN for addressing several complex real-life problems.

5. SUMMARY AND FUTURE WORK

We have presented a methodology to systematically integrate the data-mining search process with traditional statistical methods. We have applied data mining with the goal of refining the existing theoretical approach, whereas statistical methods were used to validate the conjectures against data. This hybrid approach allows us to exploit the recent advances in computationally efficient search techniques designed for large databases in conjunction with traditional statistical methods.

We illustrate our methodology with the help of two case studies and show how Association Rules Network can be used to systematically search for candidate theories. ARN provides a mechanism for synthesizing association rules in a structured manner. The important features of an ARN are (1) their ability to prune rules in the context of a goal, (2) the transformation of pruning mechanisms to simple graph operations and (3) its use as a basis of reasoning with the discovered association rules.

We are working on designing a graph layout algorithm specifically for ARNs. This will automate the display of ARNs and allow them to become part of a decision-making toolkit. We will also investigate how spurious rules affect the accuracy of an ARN for reasoning about a given goal node. It will also be interesting to investigate how the ARN for a given goal node varies with variations in the support and confidence thresholds used to derive the set of rules used for constructing the ARN, and if any patterns can be observed therein. Finally, the possibility of simultaneously analyzing multiple-goal nodes is another topic that deserves further investigation.

REFERENCES


[17] P. Tan, V. Kumar, and J. Srivastava, Selecting the right objective measure for association analysis, Inf Syst 29(4) (2004), 293–313.


[23] E.-H. Han, G. Karypis, V. Kumar, and B. Mobasher, Clustering based on association rule hypergraphs, in Proceedings SIGMOD Workshop Research Issues on Data Mining and Knowledge Discovery (DMKD ’97), 1997.


Statistical Analysis and Data Mining DOI:10.1002/sam