Actionable Pattern Mining - A Scalable Data Distribution Method Based on Information Granules

Arunkumar Bagavathi, Abhishek Tripathi, Angelina A.Tzacheva, Zbigniew W.Ras
Department of Computer Science
University of North Carolina at Charlotte
Charlotte, NC, 28223, USA
Email: abagavat@uncc.edu,atripat4@uncc.edu,aatzache@uncc.edu,ras@uncc.edu

***Abstract*—Actionable patterns are a form of recommendations that gives knowledge required by the user on actions they need to undertake to attain profit or achieve goals. Action rule is one of the methods to mine actionable patterns that are hidden in a large dataset. In the modern era of big data, organizations are collecting massive amounts of data and they keep the data updated constantly, including in major domains like financial, medical, as well as social media and IoT. Classical action rules extraction models, which analyze the data in a non-distributed fashion, lacks efficiency when processing big volumes of data and thus require a distributed approach to extract rules. Major complications of discovering action rules with distributed methods include data distribution among processing nodes in the cluster and extracting mined rule parameters like: support, confidence, utility, and coverage, representing the entire data. In this work, we focus on data distribution phase of the distributed actionable pattern mining problem. Very few works have been proposed to distribute the big data in both horizontal and vertical fashions, and extract rules/knowledge from the distributed data. We primarily focus on the vertical data splitting strategy using information granules, which give meaningful representations of complex information systems as fine grained granules. We tend to make the vertical data partitioning more logical using information granules instead of splitting the data in random.**

***Keywords*-Action Rules, Information granules, Vertical data distribution, Spark**

I. Introduction

Granular Computing (GrC) is a domain that makes use of information granules for solving complex human-centric problems [1]. The idea of granular computing is widely used in multiple areas like data processing, machine learning, rough set theory, decision trees and artificial intelligence. With the key idea of information granules, Granular Computing can also be used in knowledge representation and data mining. *Information granules* are a collection of granules, where each *granule* is a set of data objects are stacked together based on their similarity, functionality or indistinguishability [2]. Thus a granule can be seen as a subset of a larger problem, that can be used effectively to solve a complex task. Information granulation is the process of breaking a complex object into smaller pieces called information granules. Information granulation, thus can solve more complex problems by considering meaningful levels of granularity of the problem [3].

Discovering surprising, unknown and useful knowledge from a massive data in the crucial task of data mining. Most of the data mining or machine learning algorithms manifest the relationship of data objects with other objects (Clustering) or classes (Classification). The Rule based learning tasks intend to circumscribe methods that identifies, learns or evolves 'rules' to store and manipulate knowledge. In the field of data mining, many rule based systems like Association Rules and Decision Rules exist to generate rules to associate patterns and for classification respectively. Rules takes the format as given in Equation 1, where the *antecedent*(left side of the rule) is a conjunction of conditions and the *consequent* (right side of the rule) is a resulting pattern for the conditions in antecedent.

$$condition(s) \rightarrow result(s) \tag{1}$$

Action Rule is a rule based data mining technique that recommend possible transitions of data from one state to another, which the user can use to their advantage. For example, improving customer satisfaction in business [4] and reducing hospital readmission in the medical domain [5]. Action Rules are extracted from Decision table [6]. Action Rules can take the representation as given in Equation 2, where Ψ represents a conjunction of stable features, $(\alpha \rightarrow \beta)$ represents a conjunction of changes in values of flexible features and $(\theta \rightarrow \phi)$ represents desired decision action.

$$[(\Psi) \wedge (\alpha \rightarrow \beta)] \rightarrow (\theta \rightarrow \phi) \tag{2}$$

More than a decade, many research have been conducted over Action Rules extraction giving rise to several algorithms like DEAR [7], ARAS [8] and Association Action Rules [9]. These algorithms extract knowledge efficiently when the data is small, which is not the case in this era of big data. Limited research like MR-Random Forest[10] and SARGS [11] has been done on extracting Action Rules in a distributed scenario. The ultimate challenges in extracting Action Rules in a distributed fashion is that distribution of

data among the computation nodes has to be done in such a way that there is minimum loss of actionable knowledge extracted from the distributed data. In this paper, we propose a granularity based method to handle the data distribution task and extract Action Rules efficiently using a popular distributed computing called Spark [12]. We also apply our algorithm to datasets in the domains of transportation, medical and business.

II. Related Works

Although Granular computing was proposed by Zadeh [2] purely to represent human cognition, the idea of the topic has been adopted in multiple problems like decision trees [13], divide and conquer [14], set theory, data reduction or compression, discretization [15], etc. One of the applications of information granules are finding optimal solution that satisfies certain imprecise human assigned conditions [14]. Granular computing has been used in various image processing applications, by grouping some pixels into semantically meaningful clusters or granules[16]. Recently, Liu, et.al [17] used information granules for time series models.

Rule based systems are used popularly in multiple machine learning methods like classification, regression and association [18]. Rule based systems can be used for searching, extraction and representation of knowledge find great use of information granules to do their tasks efficiently [13]. Liu, et.al [19] considers each rule from the learning algorithm as a granule and use such granules for rule generation, rule simplification and rule representation. Ahmad and Pedrycz [20] used information granules to reduce the number of rules given by a recommendation system, which can be helpful in evaluating and representing rules.

Action Rules originated to help many businesses to gain profit by finding interesting actionable patterns in the data [6]. In the literature, Action Rules are extracted using two methods. First method is a rule based approach, in which intermediate classification rules are extracted first using efficient rule generation algorithms such as LERS or ERID. From these extracted rules, action rules are generated with systems like DEAR [7], which extracts Action Rules from two classification rules, or ARAS [8], which extracts Action Rules using a single classification rule. Second method is object-based approaches, in which the Action Rules are extracted directly from the decision table without any intermediary steps. Systems ARED [21] and Association Action Rules [9] works in the object-based approach. Algorithms, except association action rules, runs much faster with the aim of extracting rules that are benefits the user to the maximum and extracts only limited recommendations.

Considering the growth of big data, some research [10] [11] have been done to extract Action Rules using popular distributed computing frameworks such as MapReduce[22] and Spark[12]. [10] proposed a method to distribute the data in random to multiple sites, combining results from all sites and take an average on parameters like Support and Confidence. [11] handle the load balancing by uniformly distributing the data into partitions based on the decision attribute. Although some Action Rules extraction methods are using some form of information granules, no interesting technique were proposed on distributing the data to multiple nodes.

In this paper, we propose a novel approach for partitioning the given data using information granules. We also give a new algorithm to extract all Action Rules, based on the algorithm proposed in [9], which is the slowest among all proposed Action Rules algorithms and compute additional parameters like Utility and Coverage.

Table I
Example Decision System $\mathbb{T}$

X	A	B	C	D
x_1	Y	N	N	D_1
x_2	Y	H	Y	D_2
x_3	Y	H	Y	D_1
x_4	N	N	N	D_2
x_5	N	H	N	D_1
x_6	N	N	Y	D_2
x_7	N	H	Y	D_2
x_8	N	H	N	D_1

III. Background Knowledge

In this section, we give basic knowledge about Decision system, Action Rules, Spark and GraphX frameworks to understand out methodology.

A. Decision System

Consider a decision system given in Table I

Information System can be represented as $\mathbb{T} = (\mathbb{X}, \mathbb{A}, \mathbb{V})$ where,

X is a nonempty, finite set of objects: $\mathbb{X} = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$

A is a nonempty, finite set of attributes: $\mathbb{A} = A, B, C, D$ and

V_i is the *domain* of attribute *a* which represents a set of values for attribute $i | i \in \mathbb{A}$. For example, $V_B = N, H$.

An information system becomes Decision system, if $A = A_{St} \cup A_{Fl} \cup d$, where ***D*** is a *decision attribute*. The user chooses the attribute ***d*** if they wants to extract desired action from $d_i : i \in V_d$. A_{St} is a set of *Stable Attributes* and A_{Fl} is a set of *Flexible Attributes*. For example, *ZIPCODE* is a Stable Attribute and *User Ratings* can be a Flexible Attribute.

Lets assume from Table I that ***C*** $\in A_{St}$. A, B $\in A_{Fl}$ and ***D*** $\in d$. and the decision maker desires Action Rules that triggers the decision attribute change from D_1 to D_2 throughout this paper for examples.

B. Action Rules

In this subsection, we give definitions of action terms, action rules and properties of action rules [6]

Let $\mathbb{T} = (\mathbb{X}, \mathbb{A} \cup d, \mathbb{V})$ be a decision system, where *d* is a decision attribute and $\mathbb{V} = \cup V_i : i \in \mathbb{A}$. Action terms can be given by the expression of $(m, m_1 \rightarrow m_2)$, where $m \in A$ and $m_1, m_2 \in V_m$. $m_1 = m_2$ if $m \in A_{St}$. In that case, we can simplify the expression as (m, m_1) or $(m = m_1)$. Whereas, $m_1 \neq m_2$ if $m \in A_{Fl}$

Action Rules can take a form of $t_1 \cap t_2 \cap .... \cap t_n$, where t_i is an atomic action or action term and the Action Rule is a conjunction of action terms to achieve the desired action based on attribute *D*. Example Action Rule is given below: $(a, a_1 \rightarrow a_2).(b, b_1 \rightarrow b_2) \longrightarrow (D, D_1 \rightarrow D_2)$

1) Properties of Action Rules: Action Rules are considered interesting based on the metrics such as Support, Confidence, Utility and Coverage. Higher these values, more interesting they are to the end user.

Consider an action rule $\mathcal{R}$ of form:

$(Y_1 \rightarrow Y_2) \longrightarrow (Z_1 \rightarrow Z_2)$ where,

Y is the condition part of $\mathcal{R}$

Z is the decision part of $\mathcal{R}$

Y_1 is a set of all left side action terms in the condition part of $\mathcal{R}$

Y_2 is a set of all right side action terms in the condition part of $\mathcal{R}$

Z_1 is the decision attribute value on left side

Z_2 is the decision attribute value on right side

In [6], the support and confidence of an action rule $\mathcal{R}$ is given as

$Support(\mathcal{R}) = min\{card(Y_1 \cap Z_1), card(Y_2 \cap Z_2)\}$

$Confidence(\mathcal{R}) = [\frac{card(Y_1 \cap Z_1)}{card(Y_1)}] \cdot [\frac{card(Y_2 \cap Z_2)}{card(Y_2)}]$

Later, Tzacheva et.al [23] proposed a new set of formula for calculating Support and Confidence of Action Rules. Their idea is to reduce complexities in searching the data several times for Support and Confidence of an Action Rule. The new formula are given below.

$Support(\mathcal{R}) = \{card(Y_2 \cap Z_2)$

$Confidence(\mathcal{R}) = [\frac{card(Y_2 \cap Z_2)}{card(Y_2)}]$

Tzacheva et. al [23] also introduced a notion of utility for Action Rules. Utility of Action Rules takes a following form. For most of cases Utility of Action Rules equals the Old Confidence of the same Action Rule.

$Utility(\mathcal{R}) = [\frac{card(Y_1 \cap Z_1)}{card(Y_1)}]$

Coverage of an Action Rule means that how many decision from values, from the entire decision system S, are being covered by all extracted Action Rules. In other words, using the extracted Action Rules, *Coverage* defines how many data records in the decision system can successfully transfers from Z_1 to Z_2

C. Spark

Spark [12] is a framework that is similar to MapReduce [22] to process large quantity of data in a parallel fashion. Spark introduces a distributed memory abstraction strategy named Resilient Distributed Datasets(RDD) that can do in-memory computations on nodes distributed in a cluster. Results of each operation are stored in memory itself, which can be accessed for future processes and analyses, which in-turn creates another RDD. Thus, Spark cuts-off large number of disk accesses for storing intermediate outputs like in Hadoop MapReduce. Spark functions in two stages: 1. *Transformation*, 2. *Action*. During the *Transformation* stage, computations are made on data splits and results are stored in worker nodes memory as RDD. While the *Action* stage on an RDD collect results from all workers and send it to the driver node or save the results to a storage unit. With RDDs Spark helps machine learning algorithms to skip innumerable disk access during the iterations.

IV. Methodology

A. Data distribution strategy based on granules

The basic advantage of information granularity is that we can break bigger problems into fine grained granules. Since our problem is with distribution of data, we incorporate information granules to our method. Algorithm 1 gives a brief description about the process we use to measure overlaps between 2 granules and sub granules in each granules.

Algorithm 1 Granule Based Data Distribution

Require: partitions, dataSplit1, dataSplit2
$split1Sum \leftarrow 0.0$
for data1 in dataSplit1 **do**
  $subpartitions \leftarrow$[ ]
  $subpartitionsCount \leftarrow 0$
  for data2 in dataSplit2 **do**
    $commonLines \leftarrow data1.lines \cap data2.lines$
    if $commonLines \neq \emptyset$ **then**
      subpartitions.addAll(commonLines)
      $subpartitionsCount+ = 1$
      if $|subpartitions| == |data1.lines|$ **then**
        $split1Sum+ = 1/subpartitionsCount$
        break
$split2Sum \leftarrow 0.0$
for data2 in dataSplit2 **do**
  $subpartitions \leftarrow$[ ]
  $subpartitionsCount \leftarrow 0$
  for data1 in dataSplit1 **do**
    $commonLines \leftarrow data1.lines \cap data2.lines$
    if $commonLines \neq \emptyset$ **then**
      subpartitions.addAll(commonLines)
      $subpartitionsCount+ = 1$
      if $|subpartitions| == |data2.lines|$ **then**
        $split2Sum+ = 1/subpartitionsCount$
        break
$split1Avg = split1Sum/|dataSplit1|$
$split2Avg = split2Sum/|dataSplit2|$
return $split1Avg - split2Avg$

Table II
SUB-GRANULES OF GRANULES: *A,B* AND *C,D*

A,B	C,D
Y, N - $\{x_1\}$	N, D_1 - $\{x_1, x_5, x_8\}$
Y, H - $\{x_2, x_3\}$	Y, D_2 - $\{x_2, x_6, x_7\}$
N, N - $\{x_4, x_6\}$	Y, D_1 - $\{x_3\}$
N, H - $\{x_5, x_7, x_8\}$	N, D_2 - $\{x_4\}$

By granules, we mean a finite set of attributes from the attribute set *A* from the information system. For our initial experiments, we examine all combinations of 2 granules from the information system and minimize the correlations of granules given by Equation 3, where $C(G)$ represents correlation of a sub-granule with sub-granules of the other granule and *m,n* represents number of combinations of values of granules 1 and 2 respectively.

$$\frac{\sum_{i=1}^{m} C(G_i) + \sum_{j=1}^{n} C(G_j)}{2} \tag{3}$$

We now give an example on our optimization process for the given Information System $\mathbb{T}$ in Table I. Since we are handling data partitioning, we are taking attribute types(*Stable*,*Flexible* and *Decision* attributes) into consideration. Given an information system $\mathbb{T}$, we run our optimization(*minimizing Equation 3*) on all granules: ($\{A,B\}$ and $\{C,D\}$,$\{A,C\}$ and $\{B,D\}$ ...). Example of such combinations and sub-granules are given in Table II. In the given example, the number of sub-granules, $m, n = 4$. We measure the correlation of each sub-granule($C(G_i)$,$C(G_j)$) by checking the overlap count of the sub-granule with sub-granules of other granule. For example, $C(G_{Y,H}) = \frac{1}{2}$, since *Y,H* from *A,B* overlaps with Y, D_2 and Y, D_1 of the granule *C,D*.

The complete optimization is handled in a distributed fashion using the Spark framework [12] and getting scores of each granule are done much efficiently. A brief description about our optimization process has been given in Figure 1.

B. Scalable association action rules extraction

In this work, we follow a parallelized version of Association Action Rules [9] extraction technique in contrast to the parallel Action Rules extraction technique followed by horizontal data splitting approach [11]. Association Action Rules extraction is an exhaustive A-Priori based method of Association Rules extraction [24], which extracts the complete set of Action Rules by taking combinations of action terms through an iterative procedure. A-Priori algorithm [25] is used mostly in market basket analysis to trace transactions that appears together at great frequency. Association Action Rules method does not scale very well with high dimensional data and lacks efficiency in running time. In this work, we create partitions by splitting the data by attributes in a high dimension data as given in section IV-A. We perform Association Action Rule extraction algorithm on each of those partitions in parallel, which allows for much faster computational time for Association Action Rules extraction in Cloud platforms.

Figure 2 presents an example vertical data partitioning with the sample Decision system in Table I. Our algorithm runs separately on each partition, does transformations like *map()*, *flatmap()* functions and combine results with *join()* and *groupBy()* operations. Algorithm 2 gives our new algorithm to extract all possible Action Rules from the data in a parallel fashion.

Algorithm 2 ActionRulesExtract

Require: data of type $(r_{id}, r_{values}))$ and d_{FROM}, d_{TO} values
procedure MYPROCEDURE
2: $d_A := (s \in r | r \in data | (s, r_{id})).groupByKey()$
$c_{OLD} \leftarrow d_A$
4: $i \leftarrow 2$
parallel:
6: **while** i $\neq$ n **do**:
$c \leftarrow data.flatmap(r => (comb(r_{values}, i)), r_{id})$
8: $c_{NEW} \leftarrow c.groupByKey()$
$c_{VALID} \leftarrow c_{NEW}.filter()$
10: $c_{FROM} \leftarrow c_{VALID}.filter(d_{FROM})$
$c_{TO} \leftarrow c_{VALID}.filter(d_{TO})$
12: **if** $c_{FROM} = \emptyset$ or $c_{TO} = \emptyset$ **then break**
$atomic \leftarrow c_{FROM}.jon(c_{TO}).filter()$
14: $action_{supp} \leftarrow (r \in atomic — findSupp(r)).filter()$
if $action_{supp} = \emptyset$ **then break**
16: $atomic_{FROM} \leftarrow atomic.filter()$
$atomic_{TO} \leftarrow atomic.filter()$
18: $a_{FROM} \leftarrow atomic_{FROM}.join(c_{OLD})$
$a_{TO} \leftarrow atomic_{TO}.join(c_{OLD})$
20: $action_{conf} \leftarrow a_{FROM}.join(a_{TO})$
actions $\leftarrow action_{supp}.join(action_{conf})$
22: collect *actions*
$c_{OLD} := c_{NEW}$
24: $i := i + 1$

Our algorithm gets the pre-processed data (r_{id}, r_{values}) as input, where r_{id} is the *row id* and r_{values} is a list of values for each record in the data. The algorithm also takes **decision from** (d_{FROM}) and **decision to** (d_{TO}) values as parameters. In *Step 2* d_A gets all distinct attribute values and their corresponding data row indexes.

Finding Support and Confidence is an iterative procedure. It takes **O(nd)** times to find for all Action Rules, where **n** is the number of Action Rules and **d** is the number of data records. To reduce this time complexity, we store a set of data row indexes of each attribute value in a Spark RDD. In the first iteration, the d_A RDD acts as a seed for all

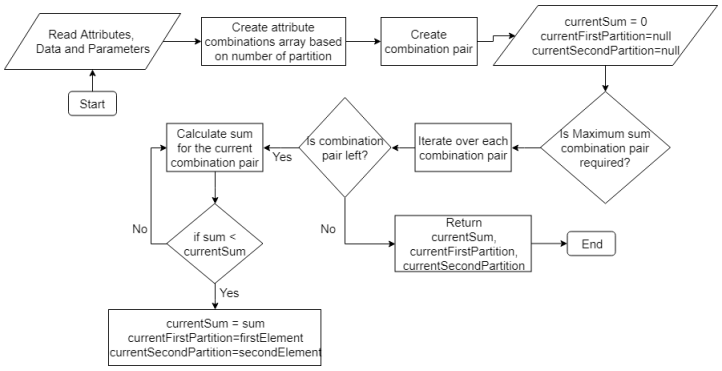

Figure 1. Data Distribution Strategy based on information granules

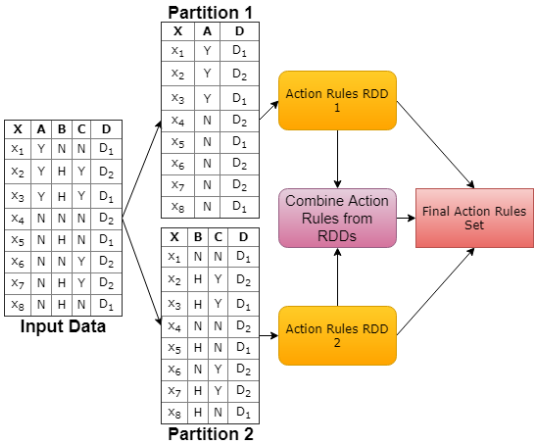

Figure 2. Example Vertical Data Distribution for Table 1

following transformations. The algorithm runs maximum of *(n)* iterations, where *n* is the number of attributes in the data. During the i^{th} iteration, the algorithm extracts Action Rules with *i-1* action set pairs on the left hand side of the rule. *Step 8* uses *flatMap()* transformation on the data to collect all possible *i* combinations from a data record. We sort the combination of attribute values since they act as key for upcoming *join()* and *groupBy()* operations. We also attach a row id r_{id} to all combinations to get the support (which data records contains a particular pattern) of each combination with the use of Spark's *groupByKey()* method in *Step 9*. In *Step 10*, we perform sequential filtering to find combinations for which the indexes count is less than the given support threshold *supp*. From the filtered combinations, we get combinations (c_{FROM} and c_{TO}) that has decision from d_{FROM} value and decision to d_{TO} values in *Step 11* and *Step 12* respectively. In *Step 14*, we join c_{FROM} and c_{TO} based on attribute names, filter outd_{FROM} and d_{TO} values since we know the decision action, which is not required in finding confidence of Action Rules.

We then calculate actual support of the resultant Action Rules and filter out rules that has support atleast the given support threshold *supp* in *Step 15* and reformat it to the form given in Equation 2. Steps *14-21* are just used to calculate confidence of an Action Rule from collected support values of the rule. We now join Action Rules with Support from *Step 15* and Action Rules with Confidence from *Step 21* to get final set of Action Rules. In *Step 23*, we assign new combination(c_{NEW}) to the old combinations(c_{OLD}) and pass the same to the next iteration.

V. Experiments and Results

To test our methods, we use three datasets: *Car Evaluation* data [26], *Mammographic Mass* data [26], and the Net Promoter Score dataset data [27].

The Car Evaluation and Mammography are obtained from the Machine Learning repository of the Department of Information and Computer Science of the University of California, Irvine [26]. The Car Evaluation Data consists of records describing a car's goodness and acceptability. Mammographic is the most effective method for screening breast cancer. Since these datasets are relatively small in size, in order to test them for scalability with the proposed distributed processing algorithms, we replicate their data rows 1024 and 2056 times respectively for CarEvaluation and MammographicMass datasets, in order to increase data size. For the *Car Evaluation* data, we choose *Class* attribute as a decision attribute and we collect Action Rules to help the car company to change the car from *Unacceptable* state to *Acceptable* state. For the *Mammographic Mass* data, we choose *Cancer Severity* as a decision attribute and we collect Action Rules to reduce the severity from *Malignant* to *Benign*.

The NPS (Net Promoter Score) [27] dataset is collected customer feedback data related to heavy equipment repair. The entire dataset consists of 38 companies and 340,000 customer surveys across sites from USA and Canada. All customer surveys are saved into database with each question (benchmark) as one feature in the dataset. Benchmarks include numerical scores (0-10) representing the quality of service: e.g. if job done correctly, are you satisfied with the job, likelihood to refer, etc. The decision attribute in the dataset is $PromoterStatus$ which labels each customer

Table III
DATASET PROPERTIES

Property	Car Evaluation Data	Mamm. Mass Data	NPS Data: Company 16	NPS Data: Company 16_31	NPS Data: Company 17	NPS Data: Company 30
Attributes	7 attributes -Buying -Maintenance -Doors -Persons -Luggage Boot -Safety -Class	6 attributes -BI-RADS -Patient's age -Shape -Margin -Density -Severity	20 attributes including - Client Name - Division - Benchmark: Dealer communication - Benchmark: Overall satisfaction - Benchmark: Technician communication	23 attributes including - Survey Type - Survey Name - Benchmark: Order accuracy - Benchmark: Personnel knowledge - Benchmark: Time taken to place order	22 attributes including - Channel Type - Division - Benchmark: Ease of ordering - Benchmark: Parts availability - Benchmark: Referral behavior	23 attributes including - Channel type - Survey name - Benchmark: Ease of using online store - Benchmark: Likelihood to be a repeat customer - Benchmark: How orders are placed
Decision attribute values	Class (unacc, acc, good, vgood)	Severity (0 - benign, 1-malignant)	Estimated Sales (Detractor Passive Promoter)			
# of instances / decision value	unacc - 1210 acc - 384 good - 69 vgood - 65	0 - 516 1 - 445	Detractor - 61 Passive - 180 Promoter - 939	Detractor - 65 Passive - 190 Promoter - 1806	Detractor - 22 Passive - 61 Promoter - 459	Detractor - 94 Passive - 391 Promoter - 2821
Replication Factor	1024	2048	N/A			
# of instances	1,769,472	1,968,128	1180	2078	547	3335

as either $promoter, passive$ or $detractor$. The decision problem here is to improve customer satisfaction / loyalty as measured by Net Promoter Score. The goal of applying action rules to solve the problem is to find minimal sets of actions so that to "reclassify" customer from "Detractor" to "Promoter" and the same improve NPS. For our experiments, we used survey given by customers for 4 companies over the year of 2015. Table III gives an overview of some properties of all datasets that we used to test our methods.

Table IV
PARAMETERS USED IN ALL ACTION RULE DISCOVERY ALGORITHMS

Property	Car Evaluation Data	Mamm. Mass Data	NPS Data
Stable attributes	Maintenance, Buying Price, Doors	Age	Survey name Survey type Division Channel type Client name
Required decision action	(Class) $unacc \rightarrow acc$	(Severity) $1 \rightarrow 0$	Promoter Status $Detractor \rightarrow Promoter$
Minimum Support α and Confidence β	2048, 70%	4096, 70%	2, 80%

In Table V, we give Action Rules extracted from all datasets used in experiments. These Action Rules provide actionable recommendations to users who wants to achieve the desired decision action. For example, from Table V, when a user wants to achieve the action $Class, unacc \rightarrow acc$, our system give actionable recommendations to achieve that goal. Action Rules AR_{C1} and AR_{C2} in Table V are example recommendations given by our system for the appropriate parameters given in Table IV. For example, Action Rule AR_{C1} recommends if the car company maintains *Buying cost* to *medium* and *Maintenance Cost* to *Very high* and decrease the *Seating capacity* from *More than 4* to *4* and increase *Safety measures* from *medium* to *high* with support of 1107 and minimum confidence of 74%

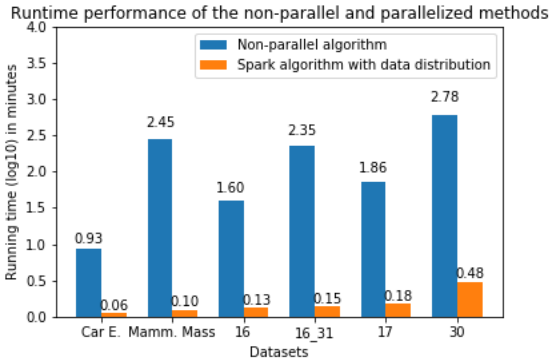

Figure 3. Speed Performance of Non-parallel and Parallel algorithms of Association Action Rules extraction

Similarly, due to space constraints, we give only sample Action Rules for all NPS datasets: *16, 16_31, 17 and 30* in Table V. In all cases, we use parameters given in Table IV. For example, consider AR_{N1} which recommends that if the company can improve user's ratings on *Completion of repair correctly* from *5 points* to *10 pints* and improve user's ratings

Table V
ACTION RULES OF CAR EVALUATION AND MAMMOGRAPHIC MASS DATASETS

Car Evaluation Data

1) AR_{C1} : $(buying = med) \wedge (maint = vhigh) \wedge (persons, more \rightarrow 4) \wedge (safety, med \rightarrow high) \Longrightarrow (class, unacc \rightarrow acc)[Support : 1107, OldConfidence : 74\%, NewConfidence : 100\%, Utility : 74\%$
2) AR_{C2} : $(buying = med) \wedge (maint = vhigh) \wedge (persons, more \rightarrow 4) \wedge (safety, med \rightarrow high) \Longrightarrow (class, unacc \rightarrow acc)[Support : 1353, OldConfidence : 85\%, NewConfidence : 100\%, Utility : 85\%$

Mammographic Mass Data

1) AR_{M1} : $(BIRADS, 6 \rightarrow 2) \wedge (Density, 4 \rightarrow 3) \Longrightarrow (Severity, 1 \rightarrow 0)[Support : 12288, OldConfidence : 100\%, NewConfidence : 100\%, Utility : 100\%]$
2) AR_{M2} : $(Age = 42) \wedge (Density, 1 \rightarrow 3) \wedge (Shape = 1) \Longrightarrow (Severity, 1 \rightarrow 0)[Support : 10240, OldConfidence : 100\%, NewConfidence : 100\%, Utility : 100\%]$

Business Datasets

1) AR_{N1} : $(Benchmark : Service - RepairCompletedCorrectly, 5 \rightarrow 10) \wedge (Benchmark : Service - TechnicianCommunication, 3 \rightarrow 9) \Longrightarrow (PromoterStatus, Detractor \rightarrow Promoter)[Support : 2.0, OldConfidence : 90.0\%, NewConfidence : 90.0\%, Utility : 100.0\%]$
2) AR_{N2} : $(BenchmarkPartsOrderAccuracy, 3 \rightarrow 10) \wedge (ClientName = HOLTCAT) \rightarrow (Division = Parts) \wedge (BenchmarkPartsTimeitTooktoPlaceOrder, 4 \rightarrow 9) \Longrightarrow (PromoterStatus, Detractor \rightarrow Promoter)[Support : 2.0, OldConfidence : 80.0\%, NewConfidence : 100.0\%, Utility : 80.0\%]$

Table VI
PERFORMANCE, IN TERMS OF NUMBER OF RESULTING ACTION RULES, USING PARALLEL AND NON-PARALLEL VERSIONS OF ASSOCIATION ACTION RULES EXTRACTION METHODS FOR ALL DATASETS

Dataset	Non-parallel algorithm	Parallel algorithm
Car Evaluation data	3500	3496
Mammographic Mass data	5790	5756
NPS data: 16	900	798
NPS data: 16_31	83643	83000
NPS data: 17	37256	37000
NPS data: 30	184000	182000

on *Technician communication* from *3 points* to *9 points*, the company can convert some *Detractors* to *Promoters* with support of 2.0 and confidence of 90.0%.

In Figure 3, we give run time analysis of Association Action rules extraction methods implemented in non-parallel method and our technique of splitting the data by attributes using information granules and extracting Association Action rules in parallel using Apache Spark framework. As mentioned earlier, since Association Action rules methodology is a complex algorithm, we can see fro the Figure 3 that non-parallel method takes long time to give actionable recommendations. On the other hand, our method takes only fraction of minutes. The speed increases when the data set size increases. For example, for NPS data: company 30, which is the largest in our datasets, the non-parallel version takes around an 60 minutes to produce results, while our method takes 3 minutes.

In Table VI, we compare number of Action Rules extracted by our parallel and non-parallel algorithms. Due to the data partitioning step involved in our method, we lose some Action Rules. In Table VII, we give Coverage measure of Action Rules extracted for each dataset and compare them with Coverage of Action Rules extracted using non-parallel methods. In 50% of our experiments we are able to achieve the same coverage as Action Rules extracted using non-parallel approach. From this table, we can assure that our method of extracting Action Rules using the distributed computing framework like Spark [12] can produce results in a faster runtime by losing only a limited knowledge.

VI. CONCLUSION

Action Rules are recently being used in variety of domains like medicine, business and natural language processing. A distributed approach to derive Action Rules from the given data can benefit many such domains. In the presented work, we proposed a novel method that divides the data using information granules rather than performing random distribution. This method provides more coherent optimization for data partitioning by taking correlations of granules with other granules. Combining this data partitioning method with Action Rules extraction produces higher quality rules, with good processing time for larger datasets. The downside of the proposed approach can be taking combinations of rules from multiple partitions, which can become complex when each partition produce large number of rule. Thus the proposed method can be improved in the future by designing an approximation function for vertical data partitioning. The data partitioning can be done in such a way that we simply merge all rules from multiple partitions instead of combining or taking cartesian product of rules.

REFERENCES

[1] J. T. Yao, A. V. Vasilakos, and W. Pedrycz, "Granular computing: perspectives and challenges," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1977–1989, 2013.

[2] L. A. Zadeh, "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic," *Fuzzy sets and systems*, vol. 90, no. 2, pp. 111–127, 1997.

[3] W.-Z. Wu, Y. Leung, and J.-S. Mi, "Granular computing and knowledge reduction in formal contexts," *IEEE transactions on knowledge and data engineering*, vol. 21, no. 10, pp. 1461–1474, 2009.

[4] J. Kuang, A. Daniel, J. Johnston, and Z. W. Raś, "Hierarchically structured recommender system for improving nps of a company," in *International Conference on Rough Sets and Current Trends in Computing*. Springer, 2014, pp. 347–357.

Table VII
PERFORMANCE OF ALGORITHMS FOR ALL DATASETS IN TERMS OF A COVERAGE MEASURE

Dataset	Non-parallel algorithm	Parallel algorithm - Random distribution	Parallel algorithm - Granule distribution
Car Evaluation data	99.5%	99%	99%
Mammographic Mass data	94%	92.53%	92.53%
NPS data: 16	89.2%	86.9%	86.9%
NPS data: 16_31	73%	70.77%	70.77%
NPS data: 17	72.7%	72.7%	72.7%
NPS data: 30	75.5%	75.5%	75.5%

[5] M. Al-Mardini, A. Hajja, L. Clover, D. Olaleye, Y. Park, J. Paulson, and Y. Xiao, "Reduction of hospital readmissions through clustering based actionable knowledge mining," in *Web Intelligence (WI), 2016 IEEE/WIC/ACM International Conference on*. IEEE, 2016, pp. 444–448.

[6] Z. W. Ras and A. Wieczorkowska, "Action-rules: How to increase profit of a company," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2000, pp. 587–592.

[7] L.-S. Tsay* and Z. W. Raś, "Action rules discovery: system dear2, method and experiments," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 17, no. 1-2, pp. 119–128, 2005.

[8] Z. W. Raś, E. Wyrzykowska, and H. Wasyluk, "Aras: Action rules discovery based on agglomerative strategy," in *International Workshop on Mining Complex Data*. Springer, 2007, pp. 196–208.

[9] Z. W. Ras, A. Dardzinska, L.-S. Tsay, and H. Wasyluk, "Association action rules," in *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*. IEEE, 2008, pp. 283–290.

[10] A. A. Tzacheva and Z. W. Ras, "Association action rules and action paths triggered by meta-actions," in *Granular Computing (GrC), 2010 IEEE International Conference on*. IEEE, 2010, pp. 772–776.

[11] A. Bagavathi, P. Mummoju, K. Tarnowska, A. A. Tzacheva, and Z. W. Ras, "Sargs method for distributed actionable pattern mining using spark," in *2017 IEEE International Conference on Big Data (Big Data)*, Dec 2017, pp. 4272–4281.

[12] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 2–2.

[13] H. Liu and A. Gegov, "Collaborative decision making by ensemble rule based classification systems," in *Granular Computing and Decision-Making*. Springer, 2015, pp. 245–264.

[14] A. Skowron and J. Stepaniuk, "Modeling of high quality granules," in *International Conference on Rough Sets and Intelligent Systems Paradigms*. Springer, 2007, pp. 300–309.

[15] V. Kreinovich, "Interval computations as an important part of granular computing: an introduction," *Handbook of Granular Computing*, pp. 1–31, 2008.

[16] H. Li, L. Zhang, B. Huang, and X. Zhou, "Sequential three-way decision and granulation for cost-sensitive face recognition," *Knowledge-Based Systems*, vol. 91, pp. 241–251, 2016.

[17] S. Liu, W. Pedrycz, A. Gacek, and Y. Dai, "Development of information granules of higher type and their applications to granular models of time series," *Engineering Applications of Artificial Intelligence*, vol. 71, pp. 60–72, 2018.

[18] H. Liu, A. Gegov, and F. Stahl, "Categorization and construction of rule based systems," in *International Conference on Engineering Applications of Neural Networks*. Springer, 2014, pp. 183–194.

[19] H. Liu, A. Gegov, and M. Cocea, "Rule-based systems: a granular computing perspective," *Granular Computing*, vol. 1, no. 4, pp. 259–274, 2016.

[20] S. S. S. Ahmad and W. Pedrycz, "The development of granular rule-based systems: a study in structural model compression," *Granular Computing*, vol. 2, no. 1, pp. 1–12, 2017.

[21] S. Im and Z. W. Raś, "Action rule extraction from a decision table: Ared," in *International Symposium on Methodologies for Intelligent Systems*. Springer, 2008, pp. 160–168.

[22] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[23] S. R. A.A. Tzacheva, C.C. Sankar and R. Shankar, "Support confidence and utility of action rules triggered by meta-actions," in *proceedings of 2016 IEEE International Conference on Knowledge Engineering and Applications*, ser. ICKEA 2016. IEEE Computer Society, 2016.

[24] M. Hahsler and R. Karpienko, "Visualizing association rules in hierarchical groups," *Journal of Business Economics*, vol. 87, no. 3, pp. 317–335, 2017.

[25] S. Rathee, M. Kaul, and A. Kashyap, "R-apriori: an efficient apriori based algorithm on spark," in *Proceedings of the 8th Workshop on Ph. D. Workshop in Information and Knowledge Management*. ACM, 2015, pp. 27–34.

[26] M. Lichman, "Uci machine learning repository," Irvine, CA, USA, Tech. Rep., 2013.

[27] Z. W. Ras, K. A. Tarnowska, J. Kuang, L. Daniel, and D. Fowler, "User friendly nps-based recommender system for driving business revenue," in *International Joint Conference on Rough Sets*. Springer, 2017, pp. 34–48.