2

Knowledge Discovery on the Grid

Lamine Aouad, An Le-Khac and Tahar Kechadi
University College Dublin
School of Computer Science and Informatics
Belfield, Dublin 4
Ireland

1. Introduction

In the last few decades, Grid technologies have emerged as an important area in parallel and distributed computing. The Grid can be seen as a computational and large-scale support, and even in some cases as a high-performance support. In recent years, the data mining community have been increasingly using Grid facilities to store, share, manage and mine large-scale data-driven applications. Indeed, data mining and knowledge discovery applications are by nature distributed, and are using the Grid as their execution environment. This particularly led to a great interest of the community in distributed data mining and knowledge discovery on large Grid platforms. Many Grid-based Data Mining (DM) and Knowledge Discovery (KD) frameworks were initiated, and proposed different techniques and solutions for large-scale datasets mining. These include the ADMIRE project initiated by the PCRG (Parallel Computational Research Group) at the University College Dublin, the Knowledge Grid project at the University of Calabria, The GridMiner project at the University of Vienna, among others.

These knowledge discovery[1] frameworks on the Grid aim to offer high-level abstractions and techniques for distributed management, mining, and knowledge extraction from data repositories and warehouses. Most of them use existing Grid technologies and systems to build specific knowledge discovery services, data management, analysis, and mining techniques. Basically, this consists of either porting existing algorithms and applications on the Grid, or developing new mining and knowledge extraction techniques, by exploiting the Grid features and services. Grid infrastructures usually provide basic services of communication, authentication, storage and computing resources, data placement and management, etc. For example, the Knowledge Grid system uses services provided by the Globus Toolkit, and the ADMIRE framework uses a Grid system called DGET, developed by our team at the University College Dublin. We will give some details about the best-known DM/KD frameworks in section 2. Note that this chapter is not intended to Grid systems or the way they are interfaced with knowledge discovery frameworks. Indeed, beyond the architecture design of Grid systems, the resources and data management policies, the data integration or placement techniques, and so on, these DM and KD frameworks need

[1] *Knowledge Discovery* is a more general term that includes the *Data Mining* process. It refers to the overall knowledge extraction process.

Source: Data Mining and Knowledge Discovery in Real Life Applications, Book edited by: Julio Ponce and Adem Karahoca,
ISBN 978-3-902613-53-0, pp. 438, February 2009, I-Tech, Vienna, Austria

efficient algorithmic approaches and implementations to optimise their performance and to address the large-scale and heterogeneity issues. This chapter focuses on this aspect of data mining on the Grid. In other words, we will not discuss the actual mapping of the knowledge discovery processes onto Grid jobs or services.

Unlike centralised knowledge extraction and data mining techniques, which are quite well established, there are many challenges and issues that need to be addresses in Grid-based DM, in order to fully benefit from the massive computing power and storage capacity of the Grid. These include issues related to the global model generation using local models from different sites, the heterogeneity of local datasets which might record different feature vectors, the global validity assessment, the scaling behaviour of the distributed techniques and the knowledge representation, etc. In addition, there are many Grid related issues such as security or overheads. There exists a limited amount of literature in distributed DM on Grids. This will be briefly reviewed in the following.

For the sake of clarity, we give a brief definition of what are knowledge discovery, and the data mining process. Knowledge discovery applies a set of appropriate algorithms and mechanisms to extract and present the knowledge from a given dataset, i.e. identifying valid, novel, potentially useful, and ultimately understandable patterns in the dataset. This process generally involves three main steps:

- Data cleaning, pre-processing and transformation. This basically prepares the dataset for the data mining process. Many algorithms have input-related constraints, and some conversions are required. These include noise removal, missing data treatment, data sampling, etc.
- Data Mining. This step describes the application and parameters of a specific algorithm to search for useful patterns within the dataset. There are three basic datasets analysis tasks in data mining, namely classification, association, and cluster analysis.
- Knowledge extraction and interpretation. This presents the results, i.e. a set of patterns, in a human-readable manner on the basis of the end-user interest. Various presentation forms exist in both centralised and distributed systems. In ADMIRE, we have developed an innovative knowledge map representation well adapted for large Grids. This will be discussed in section 3.

The rest of this chapter is organised as follow. The next section presents related work in Grid-based DM. In section 3, we describe the ADMIRE project, some of its well-adapted algorithmic techniques for the Grid, and the innovative knowledge map layer for high-level knowledge representation. The next section discusses some of the fundamental issues of both the knowledge discovery field and Grid computing. Finally, concluding remarks are made in section 5.

2. Related work

This section gives a brief review of the best-known existing projects in distributed data mining and knowledge discovery on the Grid. These emerging frameworks can be roughly classified either as domain-specific or domain-independent. Most of the KD frameworks on the Grid attempt to build domain-independent systems allowing the user to express specific problems.

2.1 TeraGrid

TeraGrid (Berman F., 2001) is a discovery infrastructure combining resources at eleven partner sites to create an integrated and persistent computational and data management

resource. Understanding and making scientific contributions of terabytes and petabytes of distributed data collections via simulation, modeling, and analysis are some of the challenges addressed by the TeraGrid project. TeraGrid tackles a range of domains and data collections including biomedical images, repositories of proteins, stream gauge measurements of waterways, digital maps of the world and the universe, etc. Then, the synthesis of knowledge, through mining for instance, from these massive amounts of data is among the most challenging applications on the TeraGrid. This allows the TeraGrid to achieve its potential and enable the full use of the TeraGrid infrastructure as a knowledge Grid system.

2.2 Knowledge grid

Knowledge Grid is a distributed framework that integrates data mining techniques. In the Knowledge Grid architecture, data mining tools are integrated within Grid mechanisms and services provided by the Globus Toolkit. It aims to deal with large datasets available on the Grid for scientific or industrial applications. The Knowledge Grid project was initiated by Cannataro et al. at the University of Calabria in Italy. The architecture of Knowledge Grid is designed in such a way that specialised data mining and knowledge discovery techniques fit with lower-level mechanisms and services provided by GTK. It is composed of two layers: the *Core K-Grid* and the *High-level K-Grid*. Basically, the *Core K-Grid* layer implements basic KD services on top of generic Grid services, while the *High-level K-Grid* layer is intended to design, compose, and execute distributed KD over it.

The lower layer of knowledge Grid comprises two main services: The *Knowledge Directory service* (KDS), and the *Resource allocation and execution management service* (RAEMS). KDS manages metadata including data sources and repositories, data mining tools and algorithms, a distributed execution plans which are basically a graph describing the interactions between processes and the dataflow, and finally the results of the computation, i.e. models or patterns. There are different metadata repositories associated with these services. RAEMS is used to map the application, i.e. the graph, into available resources. This component is directly based on the GRAM services of Globus, and execution plans generate requests which are translated into the Globus RSL language.

The higher layer of KG includes services for composing, validating, and executing distributed KD computations, as well as storing, analysing, and presenting services. This is offered by four main services: the *Data Access Service* (DAS), the *Tools and Algorithms Access Service* (TASS), the *Execution Plan Management Service* (EPMS), and the *Results Presentation Service* (RPS). The first is basically used for the search, selection, extraction, transformation, and access of the datasets. The second is responsible of searching, selecting, and downloading data mining tools and algorithms. The third service generates an abstract execution plan describing the computation and the mapping onto Grid resources. The last service generates, presents and visualises the discovered models and patterns. For more details about the architecture of KG, and other aspects such as the design of applications within KG, we refer the reader to (Cannataro M. et al. 2004).

This framework is more focused in providing a distributed DM architecture that can benefit from 'standard' Grid services provided by Globus. The algorithmic aspect, i.e. well-adapted approaches for the Grid, is not taken into account. In addition, Knowledge Grid does not provide global management and coordination of the overall knowledge on the Grid. These aspects are taken into account in the ADMIRE framework which makes the knowledge discovery on distributed Grids more flexible and efficient.

2.3 GridMiner

GridMiner is a Grid and Web based architecture for distributed KD, based on the OGSA architecture. Each service in GridMiner is implemented as a Grid service specified by Open Grid Services Architecture. The data mining process within GridMiner is supported by several Grid services that are able to perform data mining tasks and OLAP (Brezany P. et al., 2005). It also provides a workflow engine (*Dynamic Service Control Engine*) which controls the Grid services composition provided as a *DSCL (Dynamic Service Control Language)* document by the *DSCE client*. GridMiner uses OGSA-DAI (*Data Access and Integration*) as a standard middleware implementation of its GDS (*Grid Data Service*) for supporting access and integration of data within the Grid. GridMiner has a GUI that offers a friendly front-end for the end-user and the system administrator.
GridMiner is quite similar to Knowledge Grid; the main difference is that the KG framework is based on a non-OGSA version of the Globus Toolkit (Version 2). This project is also an architecture-oriented effort and does not address the algorithmic aspect on the Grid as well as the high-level knowledge representation.

2.4 Discovery net

Discovery Net is a service-oriented computing model for knowledge discovery which allows the end-user to connect to and use data mining and analysis tools as well as data sources that are available on-line. The overall architecture of Discovery Net is composed of three main servers: the *Knowledge Servers* allow the user to store/retrieve or publish knowledge, the *Resources Discovery Servers* publish service definitions/locations, and the *Discovery Meta-information Server* stores information about each type of knowledge. Discovery Net also provides a composition language called *DPML* (*Discovery Process Markup Language*) representing the graph of services. Details about architectural aspects of Discovery Net can be found in (Curcin V. et al., 2001). Note that this framework focuses on remote services composition and has a centralised knowledge representation for each of the composed graph services. This approach is not feasible for large-scale and complex heterogeneous scenarios.

3. ADMIRE: a grid-based data mining and knowledge discovery framework

In this section, we present the architecture of the ADMIRE framework. The overall organisation of ADMIRE is presented in Fig. 1. We will be focusing on two fundamental parts of ADMIRE, namely the Grid-based algorithms and the knowledge map layer. We will present two lightweight algorithms: for clustering analysis and mining association rules on the Grid, as well as the concept of the knowledge map and its structure.
ADMIRE is organised on a layered architecture built on top of the DGET Grid middleware developed at the University College Dublin (Hudzia B. et al., 2005a), (Hudzia B. et al., 2005b). Details about the use of the specific Grid services provided by DGET within ADMIRE are not discussed in this chapter. Indeed, following the Grid architecture approach, the knowledge discovery services can be developed and implemented in different ways using the available Grid services. This is of little use for the understanding of the algorithmic approaches and the knowledge management within ADMIRE presented below. There are two main hierarchical levels in ADMIRE: the data mining level, and the knowledge map level. Modules and services within these levels will be described in the following.

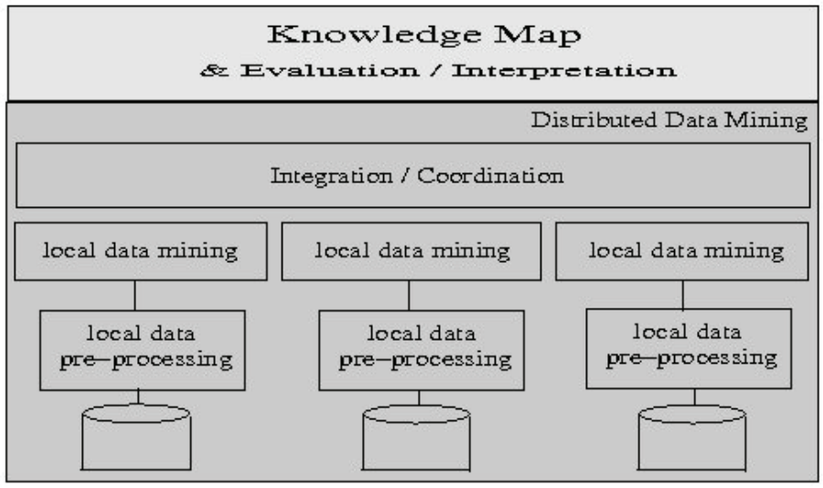


Fig. 1. The ADMIRE layered organisation.

3.1 Grid-based algorithms for large-scale datasets mining

Many parallel, distributed, and Grid-based algorithms have been already proposed in the literature for a large range of data mining applications. Most of them are based on the aggregation of local models according to some collected local information or statistics. In Grid environments, the mining algorithms have to deal with distributed datasets, different administration domains, and probably plural ownership and users. Thus, moving these datasets to a single location for performing a global mining is not always possible due to different reasons related to policies or technical choices. The Grid also faces a scalability issue of its DM applications and their implementations. We believe that the communication efficiency of an algorithm is often more important than the accuracy of its results. Indeed, communication issues are the key factors in the implementation of any distributed and grid-based algorithm. A suitable algorithm for high-speed networks is more likely to be of little use in WAN-based and Grid platforms. Efficient Grid-based algorithms need to exchange a few data and avoid synchronisations as much as possible. Following this reasoning, we proposed some well-adapted algorithms for the Grid. The rest of this section describes two of them, for clustering and frequent itemsets generation.

3.1.1 Variance-based distributed clustering

Clustering is one of the basic tasks in data mining applications. Basically, clustering groups data objects based on information found in the data that describes the objects and their relationships. The goal is to optimise similarity within a cluster and the dissimilarities between clusters in order to identify interesting patterns. This is not a straightforward task in unsupervised knowledge discovery. There exists a large amount of literature on this task ranging from models, algorithms, validity and performances studies, etc.

Clustering algorithms can be divided into two main categories, namely partitioning and hierarchical. Different elaborated taxonomies of existing clustering algorithms are given in the literature. Details about these algorithms is out of the purpose of this chapter, we refer the reader to (Jain A. K. et al., 1999) and (Xu R. & Wunsch D., 2005). Many parallel clustering versions based on sequential and centralised algorithms, such as the widely used k-means

algorithm or its variants have been proposed. These include (Dhillon I. S. and Modha D., 1999), (Ester M. et al., 1996), (Garg A. et al., 2006), (Geng H. et al., 2005), (Joshi M. N., 2003), (Xu X. et al., 1999), among others. Most of these algorithms are message-passing versions of their corresponding centralised algorithms and need either multiple synchronisation constraints between processes or a global view of the dataset or both.

Many of the proposed distributed approaches are based on algorithms that were developed for parallel systems. Indeed, most of them typically produce local models followed by the generation of a global model by aggregating the local results. The distributed processes, participating to the computation, have to be quite independent. After local phases, the global model is then obtained based on only local models, without a global view of the whole dataset. These algorithms usually perform global reduction of so-called *sufficient statistics*, probably followed by a broadcast of the results. Some research works are presented in (Januzaj E. et al., 2003), (Zhang B. and Forman G., 2000), (Januzaj E. et al., 2004a), (Januzaj E. et al., 2004b), or (Jin R. et al. 2006). These are mostly related to the k-means algorithm or its variants and the DBSCAN density-based algorithm.

There are still several open questions in the clustering process. These include:

- What is the optimal number of clusters?
- How to assess the validity of a given clustering?
- How to allow different shapes and sizes rather than forcing them into balls and shapes related to the distance functions?
- How to prevent the algorithms initialization and the order in which the features vectors are read in from affecting the clustering output?
- How to find which clustering structure for a given dataset, i.e. why would a user choose an algorithm instead of another?

These questions, among others, come from the fact that there is no general definition of what is a cluster. Indeed, algorithms have been developed to find several kinds of clusters; spherical, linear, dense, or drawnout.

In the process of addressing some of these fundamental issues, we proposed a lightweight Grid-based clustering technique, based on a merging of independent local sub-clusters according to an increasing variance constraint. This was shown to improve the overall clustering quality and finds the number of clusters and the global inherent clustering structure of the global dataset with a very low communication overhead. The algorithm finds a proper variance criterion for each dataset based on a statistical global assessment that does not violate the locality principle of this algorithm and for each dataset. This parameter can also be available from the problem domain for a given data. In the rest of this section, we will give the algorithm foundations and the complexity and performance analysis.

The algorithm foundations

The most used criterion to quantify the homogeneity inside a cluster is the variance criterion, or sum-of-squared-error. The traditional constraint used to minimize this criterion is to fix the number of clusters to an a priori known number, as in the widely used k-means and its variants (Xu R. & Wunsc D., 2005), (Ng R. T. & Han J., 1994), (Zhang B. et al., 1999), etc. This constraint is very restrictive since this number is most likely not known in most cases. Many approximation techniques exist including the gap statistic which compares the change within cluster dispersion to that expected under an appropriate reference null distribution (R. Tibshirani et al., 2000) and (Mingjin Y. & Keying Y., 2007), or the index due

to Calinski & Harabasz (Calinski R. B. & Harabasz J., 1974), among other techniques. The imposed constraint in our method states that the increasing variance of the merging, or union of two sub-clusters is below a given dynamic threshold. This parameter is highly dependent on the dataset and is computed using a global assessment method.

The key idea of the algorithm is to start with a relatively high number of clusters in local sites which are referred to as sub-clusters. An optimal local number using an approximation technique or a method that finds the number of clusters automatically, such as those described earlier, can be considered. Then, the global merging is done according to an increasing variance criterion requiring a very low communication overhead. Recall that this algorithm finds a proper variance criterion for each dataset based on a statistical global assessment. This allows us to comply with the locality criterion for different datasets.

In local sites, the clustering can be done using different algorithms depending on the characteristics of the dataset. This may include k-means, k-harmonic-means, k-medoids, or their variants, or the statistical interpretation using the expectation-maximization algorithm, etc. The merging of local sub-clusters exploits the locality in the feature space, i.e. the most promising candidates to form a global cluster are sub-clusters that are the closest in the features space, including sub-clusters from the same site. Each processing node can perform the merging and deduce the global clusters formation, i.e. which sub-clusters are subject to form together a global cluster. Fig. 2. shows how sub-clusters from different sites (Gaussian distributions) are merged together to form a global cluster.

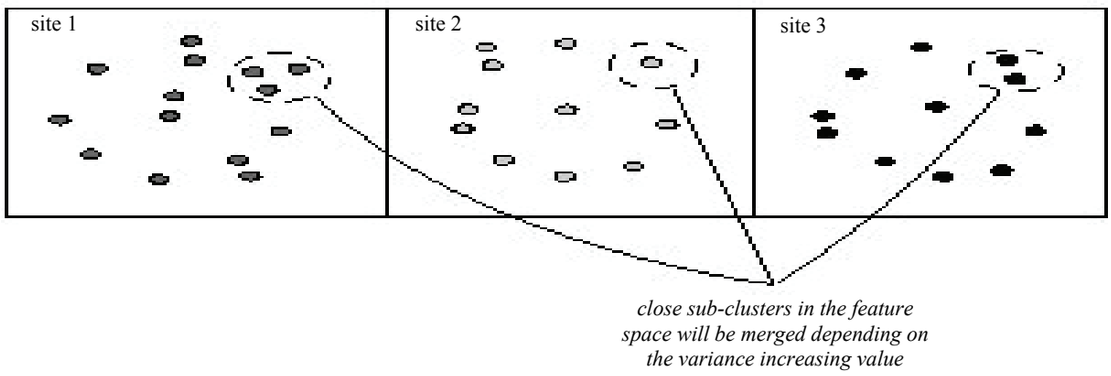


Fig. 2. Close sub-clusters in the features space are merged on a global cluster.

One important notion used in this algorithm is the global cluster border which represents local sub-clusters at its border. These sub-clusters are candidate to be moved to neighbouring global clusters in order to contribute to an improvement of the clustering output, with respect to the variance criterion, i.e. that minimises the sum-of-squared-error. These sub-clusters are referred to as perturbation candidates. The initial merging order may affect the clustering output, as well as the presence of non well-separated global clusters. This is intended to reduce the input order impact. The global clusters are then updated. The border is collected by computing the common Euclidean distance measure. The b farthest sub-clusters are then selected to be the perturbation candidates, where b depends on the local number of sub-clusters at each site and their global composition. This process naturally affects sub-clusters initially assigned to multiple global clusters.

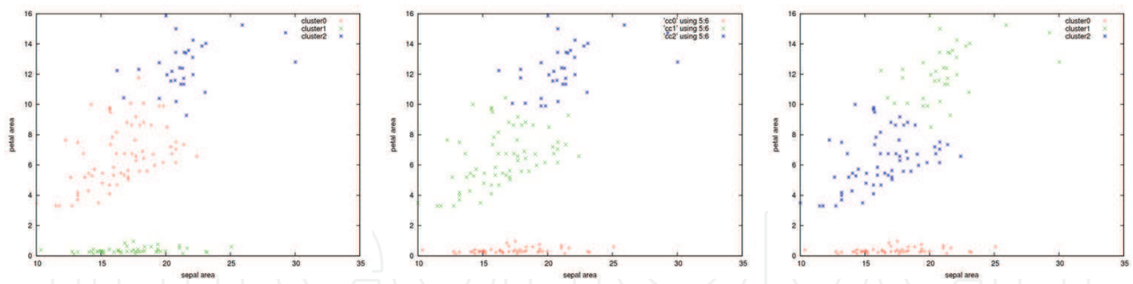
Another important aspect of this algorithm is that the merging is a labelling operation, i.e. each local site can generate the global model which is the correspondences between local sub-clusters, without necessarily reconstructing the overall clustering output. That is

because the only bookkeeping needed from the other sites is *centres*, *sizes* and *variances*. The aggregation is defined as a labelling process between local sub-clusters in each site. No datasets move is needed. On the other hand, the perturbation process is activated if the merging action is no longer applied. The perturbation candidates are collected for each global cluster from its border, which is proportional to the overall size composition. Then, this process moves these candidates by trying the closest ones and with respect to the gain in the variance criterion when moving them from the neighbouring global clusters. Formal definitions and details of this algorithm are given in (Aouad L. M. et al., 2007) and (Aouad L. M. et al. 2008b).

Complexity and performance

The computational complexity of this distributed algorithm depends on the algorithm used locally, the global assessment algorithm, the communication time which is a gather operation, and finally the merging computing time. If the local clustering algorithm is K-means for example, the clustering complexity is $O\left(N_{max}k_{max}d\right)$, where d is the number of attributes. The complexity of the global assessment depends on the size of local statistics. If the gap statistic is used on local centers, this will be $O\left(B\left(\sum k_i\right)^2\right)$, where B is the number of the reference distributions. The communication cost is $3\mathrm{d}\sum t^i_{comm}k_i$. Since k_i is much smaller than N_i, the generated communication overhead is very low.

The merging process is executed u times. This is the number of iterations until the merging condition is no longer applied. This requires $ut_{newStatistcs} = O\left(d\right)$. This is followed by a perturbation process which is of order of $O\left(bk_gk_{max}\right)$. Indeed, since this process computes for each of the b chosen sub-clusters at the border of a given cluster C_i, k_i distances for each of the k_g global clusters. The total complexity is then $O\left(dN_i\left(\sum k_i\right)^2\right)$ $(T_{comm} << O\left(N_ik_id\right))$.



(a) (b) (c)

Fig. 3. Generated clustering .using *5* (a) and *7* (b) sub-clusters, and a centralised clustering using K-HarmonicMeans in (c).

The algorithm is tested on a range of artificial and real world datasets including large Gaussian distributions, the well-known Iris dataset, the animal dataset, and the PUMS census dataset available from the UC Irvine KDD Archive. The algorithm finds the inherent number of clusters by varying the maximum variance constraint, independently of the local clustering algorithm and the number of sub-clusters. An example using the Iris dataset, where the maximum variance constraint was twice the highest individual variance, is shown in the Fig. 3. In this case, since the k-harmonic-means does not impose a variance constraint it can find a lower sum-of-squared-error locally. However, the variance-based

clustering finds the *3* initial classes based on *5* and *7* sub-clusters locally using the same increasing variance value. More experiments and evaluation are shown in (Aouad L. M. et al., 2007) and (Aouad L. M. et al. 2008b).

3.1.2 Grid-based frequent itemsets mining

The frequent itemsets mining task is at the core of various data mining applications. Since its inception, many frequent itemsets mining algorithms have been proposed (Agrawal R. & Srikant R., 1994), (Brin S. et al., 1997), (Han J. et al., 2000), (Park J. S. et al., 1995), (Savasere A. et al., 1995), among others. Many of these approaches are based on the Apriori or the FP-Growth principles. Basically, frequent itemsets generation algorithms analyse the dataset to determine which combination of items occurs together frequently. For instance, considering the commonly known market basket analysis; each customer buys a set of items representing his/her basket. The input of the algorithm is a list of transactions giving the sets of items among all existing items in each basket. For a fixed support threshold s, the algorithm determines which sets of items of a given size k are contained in at least s transactions.

The focus then is on mining frequent itemsets on distributed datasets over the Grid. Such a Grid-based approach is motivated by the challenge of developing scalable solutions for a highly computationally expensive and data intensive application. Indeed, effective distributed approaches for large-scale data mining should take into account both the challenges raised by the underlying Grid system and the complexity of the task itself. For the purpose of developing well-adapted Grid implementations, we introduce a performance study of frequent itemsets mining of large distributed datasets on the Grid, based on the widely used Apriori principle.

We studied the distributed aspect and the performance of Apriori-based approaches both theoretically and experimentally (Aouad L. M. et al., 2008). The theoretical study presents a performance model of distributed algorithms based on the Apriori principal. Note that the main factor of an Apriori-based distributed algorithm is the number of candidates generated at each step or level. This factor, which governs the algorithm complexity, can be exponential of the size of the input. Considering the case where every transaction, in every site, contains every item, the algorithm must output and may communicate each subset of the whole set of items. we show that local pruning strategies are sufficient and that global phases in classical distributions affect the performance of the system when using the Apriori principal.

The proposed approach has two main phases. The first phase consists of generating frequent itemsets on each node based only on their local datasets. This phase is the *local mining phase* and it uses the sequential Apriori algorithm. After this phase, the result will be the set of all locally frequent itemsets in each node. This information is sufficient for determining all globally frequent itemsets, using a top-down search. The second phase is the *global collection phase*. Each node broadcasts its frequent itemsets, of size k *(requested size)* and the maximal ones, to the others nodes of the system and asks for their respective support counts. The globally frequent itemsets are then identified by merging local support counts from each node. Then, the algorithm iterates on the subsets of itemsets that fail the global frequency test. More precisely the globally frequent itemsets are generated as follows:

1. Initially collect support counts of frequent itemsets of the requested size k and all smaller frequent itemsets that are not subsets of any larger frequent itemset (maximal itemsets),

2. Generate globally frequent itemsets and put all the itemsets that are not globally frequent in a set **F**.
3. If **F** is not empty, collect support counts of subsets of itemsets in **F** and go to (2).

This top-down search has been shown to be efficient in large Grids, and the overheads due to synchronisations and communications are significantly reduced. Indeed, this leads to much fewer communication passes. The global pruning steps in classical distributed approaches are computationally inefficient in local nodes and affects the global system performance.

Discussion and evaluation

Comparisons with a classical Apriori-based distributed approach, namely the Fast Distributed Mining of association rules (FDM), show that in terms of computation, both algorithms perform approximately the same amount of work as they have the same amount of candidates in the local Apriori generation. However, in terms of communication, the proposed top-down approach performs better and has only two communication passes for a range of synthetic and real datasets, namely the PUMS census dataset, and datasets generated using the IBM Quest code. The IBM Quest code is a simulation model for supermarket basket data. It has been used in several frequent itemsets generation studies such as (Han J. et al., 2000), (Purdom P. W. et al., 2004), and (Schuster A. et al., 2003), etc.

As an example, Fig. 4 shows plots of different candidates sets on different nodes using various support thresholds, on the two mentioned datasets. The lower bound, which is the ratio between the number of candidate sets of the two techniques, is *0.78*. This value is close to *1* in most cases, with and average of *0.93*. If we look at the ratio of the number of *1*-itemsets for the two techniques we can see the same behaviour with an average of *0.94*. One can conclude that the difference in terms of candidate set generation between the two

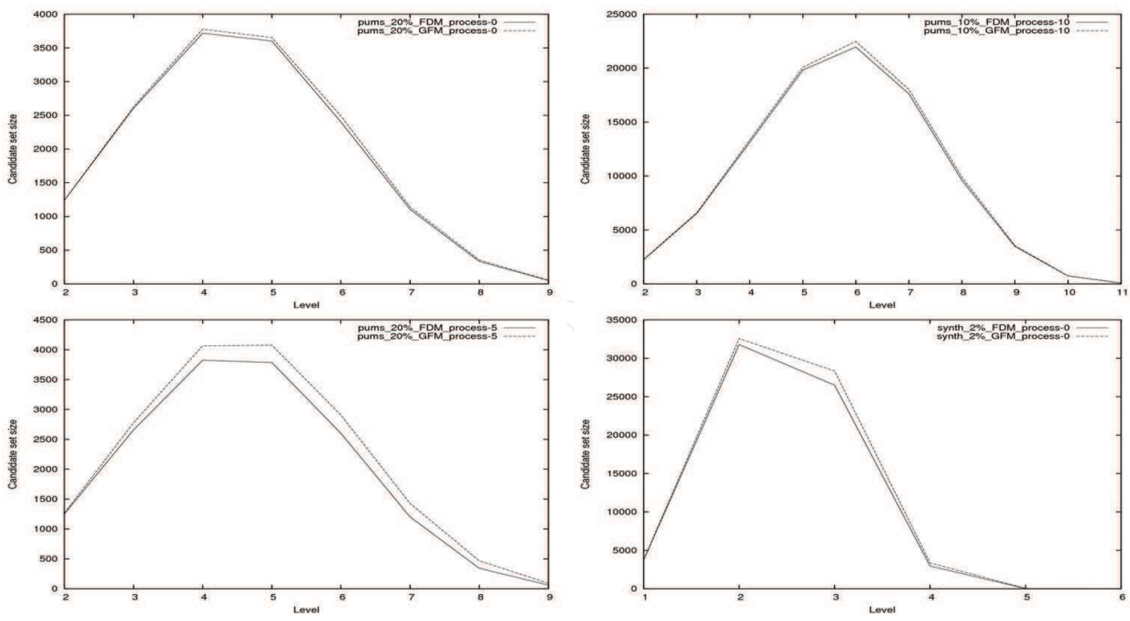


Fig. 4. Generated candidate sets using both approaches, on different processes.

techniques is not significant. In terms of processing time, this is in the order of few seconds in all cases. For the overall computation costs, the proposed technique has a gain factor of up to *82%*. However, this highly depends on the size of frequent itemsets and the number of

communication passes. Also, the inputs and outputs requirements were not considered in this model for simplicity. This is likely to be more costly for classical distributed approaches since the proposed approach generates less important overall sets for remote support count collection.
The results show that distributed implementations of the Apriori algorithm do not need global pruning strategies. Therefore, classical distributions are less efficient than the adopted global strategy in our approach, starting from the requested size and using a top-down search. Note that remote support counts computations can be very expensive in classical distribution, especially in lower levels where the numbers of locally frequent itemsets are very high. This was avoided and reduced to a minimum in the proposed approach since only a few passes of remote computations are required and with smaller sizes. Formal definitions and more detailed results are presented in (Aouad L. M. et al., 2008a). This method is intended not only to reduce synchronisation and communication overheads but also the grid tools overheads related to jobs preparation or scheduling for instance.

3.2 Knowledge map

The knowledge map concept represents what is called *the knowledge about knowledge.* This basically means the sources, structures, and representation of the acquired knowledge. Different knowledge map structures can be found in the literature including hierarchical or radial knowledge structure maps, networked knowledge maps, knowledge source maps and knowledge flow maps (Jetter A., 2006). These knowledge maps do not codify the knowledge itself, but rather guide the way and help to find the knowledge. Often, geographical maps are used as a metaphor for knowledge maps in the sense that it simplifies complex reality, downsizes it to the important aspects, and add relevant information that help to detect the way, i.e. the knowledge and its source. In the ADMIRE project, we have developed a well-adapted knowledge mapping approach for an efficient knowledge retrieval on large Grid systems. The following of this section will briefly discuss the knowledge representation in general, and then focus on the structure of our knowledge map and its evaluation.
There are different ways of representing the acquired knowledge from mined data. This includes tables, decision trees, rules-based, instance-based, clusters, etc. One of the most popular approaches to knowledge representation is production rules, also called the *if-then* rules. In cluster approaches, the output takes the form of a diagram showing how instances fall into clusters. There are many kinds of cluster representations such as space partitioning, Venn diagram, table, tree, etc. Other knowledge representation approaches, such as Petri net, Fuzzy Petri nets, or G-net were also developed and used.

3.2.1 The knowledge map structure

This section briefly describes the Knowledge Map (*KM*) structure. More details can be found in (Le-Khac N-A. et al., 2007) and (Le-Khac N-A. et al., 2008). In our context, the KM facilitates the deployment of distributed DM by supporting users' coordination and interpretation of the results. The objectives of our KM architecture are: 1) to provide an efficient way to handle a large amount of data collections in large-scale distributed systems; 2) retrieving easily, quickly, and accurately the knowledge; and 3) supporting the integration process of the results. In order to achieve these goals, KM system consists of the

following components: knowledge navigator, knowledge map core, knowledge retrieval, local knowledge map and knowledge map manager (Fig. 5).

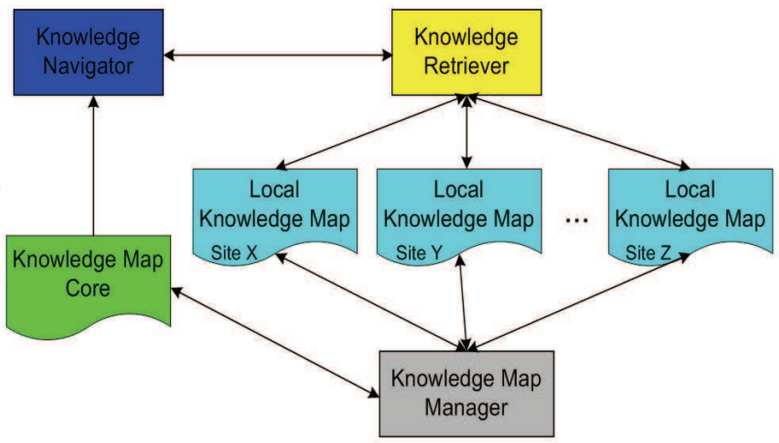


Fig. 5. Knowledge Map structure.

Knowledge navigator

The knowledge navigator component is responsible for guiding users to explore the *KM* and for determining the knowledge of interest. The result of this task is not the knowledge but its meta-data, called *meta-knowledge*, which includes related information such as data mining tasks used, data type, and a brief description of this knowledge and its location. For example, a user might want to retrieve some knowledge about tropical cyclone. The application domain "meteorology" is used by this component to navigate the user through tropical cyclone area and then a list of information related to it will be extracted. Next, based on this meta-knowledge and its application domain, the users will decide which knowledge and its location are to be retrieved.

Knowledge Map Core

This component (Fig. 6) is composed of two main parts: *concept tree repository* and it *meta-knowledge repository*. The former is a repository storing a set of application domains. Each application domain is represented by a *concept tree* that has a hierarchical structure such as a concept map (Novak J. D., 1984). A node of this tree, so called *concept node* represents a sub-application domain and it includes a unique identity in the whole *concept tree* repository and the name of its sub-application domain. The content of each *concept tree* is defined by the administrator before using *KM* system. In our approach, a mined knowledge is assigned to only one sub-application domain and this assignment is given by the user. By using *concept tree*, we can deal with the problem of knowledge context. For instance, given the distributed nature of the knowledge, some of them may have variations depending on the context in which it is presented locally.

Knowledge Retrieval

The role of this component is to seek the knowledge that is potentially relevant. This task depends on the information provided by the users after navigating through application domains and getting the meta-knowledge needed. This component is similar to a search engine which interacts with each site and collects the local knowledge.

Local Knowledge Map

This component is local to each site of the system. *Local knowledge map* is a repository of knowledge entries. Each entry, which is a knowledge object, represents a mined knowledge

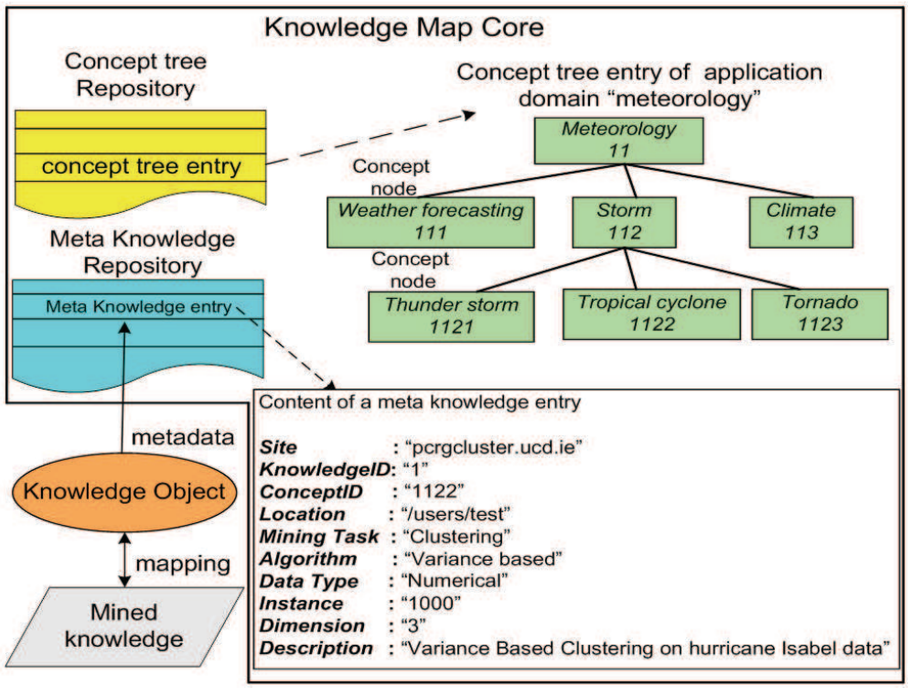


Fig. 6. Knowledge Map Core structure

and contains two parts: *meta-knowledge* and a *representative*. *Meta-knowledge* includes information such as the identity of its mined knowledge that is unique in this site, its properties, and its description. This *meta-knowledge* is also submitted to the *Knowledge map core* and will be used in *meta-knowledge entry* of its repository to be used at the global level. The *representative* of a knowledge entry depends on a given mining task. Currently, *KM* supports two kinds of representatives: one for clustering task and another for rule-based knowledge. Moreover, our system has the capacity of adding more representative types for other mining tasks.

For rule-based knowledge, the mined knowledge is represented as a set of production rules (Buchanan B. G. and Shortlife E. H., 1984). As mentioned above, a rule is of the form "if {*cause expression*} then {*conclusion expression*}" and an expression (cause or conclusion) contains a set of items. A rule also includes its attributes such as *support* and *confidence* in the association rules task or *coverage* and *accuracy* in the classification task (Han J. and Kamber M., 2006).

In the clustering case, a representative of the mined knowledge stands in one or many clusters. A cluster has one or more representative elements and each element consists of fields filled by the user. The number of fields as well as data type of each field depends on the clustering algorithm used. The meta-data of these fields is also included in each representative. A cluster also contains information about its creation. This information shows how this cluster was created: by clustering or integration process. In the former case, the information is a tuple of *{hostname, cluster filename, cluster identity}* and in the latter, it is a tuple of *{hostname, knowledge identity, cluster identity}*, where *hostname* is the location of the clustering results, which are stored in files called cluster files with their *cluster filenames*. Each cluster has a *cluster identity* and it is unique in its knowledge entry. In (Le-Khac N-A. et al., 2008), the authors describe both kinds of representatives in details.

Knowledge Map Manager

The knowledge map manager is responsible for managing and coordinating the local knowledge maps and the knowledge map core. For *local knowledge map*, this component provides primitives to create, add, delete, and update knowledge entries and their related components in knowledge repository. It also allows to submit local meta-knowledge to its repository in *knowledge map core*. This component provides also primitives to handle the meta-knowledge in the repository as well as the concept node in the concept tree repository.

3.2.3 Evaluation

The KM layer is presently being tested in the ADMIRE system. It is difficult to evaluate our approach by comparing it to other systems because it is unique so far. Therefore, this new approach is validated by evaluating different aspects of the system architecture for supporting the management, mapping, representing and retrieving the knowledge. First, we evaluate the complexity of search/retrieve the knowledge object of the system. This operation includes two parts: searching relative concept and search/retrieve the knowledge. Let N be the number of *concept tree* entries and n be the number of *concept nodes* for each *concept tree*. The complexity of the first part is $O(\log N + \log n)$ because the concept tree entries are indexed according to a tree model. However, the number of concept entries as well as of concept nodes of a concept tree is negligible compared to the number of knowledge entries. So this complexity depends strongly on the cost of search/retrieve operations. Let M be the number of meta-knowledge entries in the *KM core*, so the complexity of searching a meta-knowledge entry at this level is $O(\log M + Cs)$, where Cs is the communication cost between a node s and the host node where the meta-knowledge repository is stored. This depends on the bandwidth between two nodes and the size of the data size. The complexity of retrieving a knowledge object is the same as for the search operation. However, the retrieve operation depends on the number of knowledge entries m in the *local KM*.

Tests have been done to evaluate the search/retrieve performance. More details about these tests can be found in (Le-Khac N-A. et al., 2007) and (Le-Khac N-A. et al., 2008). Next, we estimate the performance of the *KM* architecture. Firstly, the structure of *concept tree* is based on the concept map. We can avoid the problem of semantic ambiguity as well as reduce the domain search to improve the speed and accuracy of the results. In the 1-n model (one server-n client nodes), the concept tree is implemented either only at the server node or at each client node. The client-server communication is needed when we interact with concept tree via the operations add, search, delete concept nodes or get the concept identity when adding new knowledge. In a large distributed system, this concept tree can be cached at each local node to reduce the communication cost because the numbers of operations of add/delete a concept node is very small compared to the number of search operations.

Secondly, the division of knowledge map into two main components (local and core) has some advantages: (i) the core component acts as a summary map of knowledge and it is a representation of knowledge about knowledge when combined with local *KM*; (ii) avoiding the problem of having the whole knowledge on one master node (or server), which is not feasible on very large distributed systems such as the Grid. By representing knowledge meta-data by their relationship links, the goal is to provide an integration view of these knowledge.

Finally, this approach offers a knowledge map with flexible and dynamic architecture where users can easily update the *concept tree* repository as well as meta-knowledge entries. The current index technique used in a rule representative is an inverted list. However, we can improve it without affecting to whole system structure by using other index algorithms (Martynov M. and Novikov B, 1996) or by applying compressed technique as discussed in (Zobel J. and Moffat A., 2006). Moreover, flexible and dynamic features are also reflected by mapping a knowledge to a *knowledge object*. The goal here is to provide a portable approach where knowledge object can be represented by different techniques such as an entity, an XML-based record, or a record of database, etc.

4. Discussion

Grid-based knowledge discovery has to address key issues related to the three main aspects of this area, namely the distributed algorithms for generating efficient global models, the knowledge representation and retrieval, and some specific Grid issues which raise questions on how to use a given Grid architecture and mechanisms to build algorithms and knowledge-based operations, i.e. whether high-level approaches for mining and representing knowledge are suitable for the Grid. We discussed different projects which focus, for the large part, on the architectural aspect, i.e. how to interface classical data mining and knowledge discovery operations with existing Grid technologies. In contrast, the ADMIRE framework offers more focus on scalable algorithms, and means to codify the knowledge about knowledge in large distributed Grids.

Grid algorithms are unlikely to scale if they require an extensive communication load. Indeed, straightforward distributions of many DM tasks have little choice but to exchange information between every possible pair of sites or nodes in the Grid. This is not scalable on large distributed systems. However, we might be able to decompose and/or approximate the problem and eliminate this communication needs. The notion of locality is then very important in DM and KD on the Grid. The typical way introduced in this chapter involves *local* data analysis, followed by the generation of a *global* data and knowledge model through the aggregation of the local results in different manners. For instance, many algorithms in peer-to-peer Grids use different network topologies and organisations in order to use properly the neighbourhood notion. Several algorithms have been developed to address basic problems, however multiple challenges still exist in terms of performance, accuracy, communication and scaling behaviour, convergence properties and stability for approximation techniques, privacy-preserving, and trust management, among others.

As for the knowledge-related operations, the chapter has briefly discussed the knowledge map notion as means to codify the knowledge about knowledge. Then, some of the ADMIRE's mapping operations and knowledge representation were presented. The main objective here is to codify both the knowledge and its navigation in order to improve the detection and retrieval of knowledge in large Grids. However, additional means of codification or communication can be taken into account in order to capture user-specific knowledge domains. This is part of the knowledge assessment which takes place prior to the the knowledge mapping itself. In addition, different application domains might lead to different results and could demonstrate the need of other operations and/or different underneath structure. This has to be taken into account in the future.

On the other hand, the Grid offers a large range of technologies, architectures and implementations, although standardisation works have been undertaken in recent years.

This makes it difficult to propose an *open* and *flexible* distributed and Grid-based knowledge discovery architecture that can be configured on top of various Grid middleware in a simple way. Furthermore, typical computational issues in the Grid, such as the important computing overhead, make the straightforward adaptation of classical DM infeasible. Some other inherent characteristics might not have been taken into account at the middleware level, and have to be addressed at higher levels, such as properties of the data management policies, replication, authentication, data protection and privacy, among others.

5. Conclusion

Mining large-scale datasets and the extraction, representation, and retrieving of knowledge on Grid systems still an active and challenging research area, either domain-specific or not. Several research work have been done so far including the most known projects shortly reviewed in this chapter. We also discussed different trends and focuses of these projects. Then, the motivations, design, and original aspects of ADMIRE have been presented. ADMIRE is a domain-independent solving environment that allows the user to express a problem using his/her own domain specific knowledge to build its application using basic data analysis and data mining operations. It offers lightweight distributed approaches able to perform large-scale computation to leverage the Grid in an efficient way.
ADMIRE also tackles the issue of clear and easy representation and manipulation of the knowledge by proposing its knowledge map layer. While the concept of the knowledge map itself is not new, its structure and implementation offer a novel and robust knowledge management and retrieval in large distributed Grids. In future works, we will take into account some domain-specific and real-world applications constraints and properties, in order to achieve its potential and enable the full use of the Grid for each of them.

6. References

Agrawal R. and Srikant R. (1994). Fast Algorithms for Mining Association Rules. VLDB'94: Proceeding of the 20th Int. Conf. Very Large Data Bases.

Aouad L. M., Le-Khac N-A. and Kechadi M-T. (2008). Performance Study of Distributed Apriori-like Frequent Itemset Mining. Technical report, University College Dublin, 2008.

Aouad L. M., Le-Khac N-A. and Kechadi M-T. (2008). A Multi-Stage Clustering Algorithm for Distributed Data Mining Environments. COSI 2008, Colloque sur l'Optimisation et les Systemes d'Information. Tizi-Ouzou, Algeria.

Aouad L. M., Le-Khac N-A. and Kechadi M-T. (2007). Lightweight Clustering Technique for Distributed Data Mining Applications. The 7th Industrial Conference on Data Mining ICDM 2007, Springer LNAI 4597.

Bellec J., Kechadi M-T., and Carthy J. (2005). A New Efficient Clustering Algorithm for Network Alarm Analysis. The 17th IASTED Intl. Conference on Software Engineering and Applications PDCS 2005.

Berman F. 2001. Viewpoint: From TeraGrid to knowledge grid. Commun. ACM, Vol. 44, No. 11, 2001.

Brin S. and Motwani R. and Ullman J. D. and Tsur S. (1997). Dynamic Itemset Counting and Implication Rules for Market Basket Data. SIGMOD'97: Proceedings ACM SIGMOD Int. Conf. on Management of Data.

Buchanan B. G. and Shortlife E. H. (1984) Rule-Based Expert Systems: The MYCIN Experiments of The Standford Heuristic Programming Projects Reading, MA: Addison-Wesley.

Calinski R. B. and Harabasz J. (1974). A dendrite method for cluster analysis. Communication in statistics, Vol. 3, 1974.

Campieta P., Di Martino S., Bertolotto M., Ferrucci F., and Kechadi M-T. (2007). Exploratory Spatio-Temporal Data Mining and Visualization. Journal of Visual Languages and Computing, Vol. 18, No. 3, 2007.

Dhillon I. S. and Modha D. (1999). A Data-Clustering Algorithm on Distributed Memory Multiprocessors. Workshop on Large-Scale Parallel KDD Systems, SIGKDD, 1999.

Ellahi T. N. and Kechadi M-T. (2004). Distributed Resource Discovery In Wide Area Grid Environments. LNCS on Computational Science, 3038, May 2004.

Ester M., Kriegel H.-P and Sander J. and Xu X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD), 1996.

Garg A., Mangla A., Bhatnagar V., and Gupta N. (2006). PBIRCH : A Scalable Parallel Clustering algorithm for Incremental Data. 10th International Database Engineering and Applications Symposium, IDEAS 2006.

Geng H. and Deng X. and Ali H. (2005). A New Clustering Algorithm Using Message Passing and its Applications in Analyzing Microarray Data. Proceedings of the Fourth International Conference on Machine Learning and Applications, ICMLA 2005.

Han J. and Jian Pei and Yiwen Yin (2000). Mining frequent patterns without candidate generation. Proceedings of the 2000 ACM SIGMOD int. conference on Management of Data.

Han J. and Kamber M. (2006). Data Mining: Concepts and Techniques. 2nd ed Morgan Kaufmann Publishers.

Hegarty D. F. and Kechadi M-T. (1996). Topology Preserving Dynamic Load Balancing for Parallel Molecular Simulations. ACM/IEEE Int. Conf. on Supercomputing, 1997.

Hudzia B., Kechadi M-T., and Ottewill A. (2005). TreeP: A Tree-Based P2P Network Architecture. IEEE, International Workshop on Algorithms, Models and tools for parallel computing on heterogeneous networks, HeteroPar 2005.

Hudzia B., McDermott L., Illahi T. N., and Kechadi M-T. (2005). Entity Based Peer-t o-Peer in a Data Grid Environment. The 17th IMACS World Congress Scientific Computation, Applied Mathematics and Simulation, 2005.

Jain A. K., Murty M. N. and Flynn P. J. (1999). Data Clustering: A Review. ACM Computing Surveys, Sep. 1999.

Januzaj E., Kriegel H-P., and Pfeifle M. (2003). Towards Effective and Efficient Distributed Clustering. Int. Workshop on Clustering Large Data Sets, 3rd Int. Conf. on Data Mining, ICDM 2003.

Januzaj E. and Kriegel H-P. and Pfeifle M. (2004). Scalable Density-Based Distributed Clustering. 8th European Conference on Principles and Practice Discovery in Databases, PKDD 2004.

Januzaj E. and Kriegel H-P. and Pfeifle M. (2004). DBDC: Density-Based Distributed Clustering. 9th Int. Conf. on Extending Database Technology, EDBT 2004.

Jin R. and Goswani A. and Agrawal G. (2006). Fast and Exact Out-of-Core and Distributed K-Means Clustering. Knowledge and Information Systems, Vol. 10, 2006.

Joshi M. N. (2003). Parallel K-Means Algorithm on Distributed Memor Multiprocessors. Technical report, University of Minnesota, 2003.

Le-Khac N.A. and Kechadi M-T. (2005). ADMIRE Framework: Distributed Data Mining on Data-Grid Platforms. Intl. Conference on Software and Data Technologies, ICSOFT 2006.

Le-Khac N-A., Aouad L. M. and Kechadi M-T. (2007). Knowledge Map: Toward a new approach supporting the knowledge management in Distributed Data Mining, KUI Workshop, IEEE International Conference on Autonomic and Autonomous Systems ICAS'07, Athens, Greece, 2007.

Le-Khac N-A., Aouad L. M. and Kechadi M-T. (2008). An Efficient Knowledge Management Tool for Distributed Data Mining Environments", International Journal of Computational Intelligence Research, ISSN 0973-1873, 2008.

Martynov M. and Novikov B. (1996). An Indexing Algorithm for Text Retrieval, Proceedings of the International Workshop on Advances in Databases and Information system (ADBIS'96), Moscow: 171-175.

Mingjin Y. and Keying Y. (2007). Determining the Number of Clusters Using the Weighted Gap Statistic. Biometrics, Vol. 63, No. 4, 2007.

Ng R. T. and Han J (1994). Efficient and Effective Clustering Methods for Spatial Data Mining. VLDB, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994.

Novak J.D. and Gowin D.B. (1984). Learning how to learn, Cambridge University Press.

Purdom P. W. and Van Gucht D. and Groth D. P. (2004). Average-Case Performance of the Apriori Algorithm. SIAM Journal on Computing, Vol. 33, No. 5. 2004.

Savasere A. and Omiecinski E. and Navathe S. B. (1995). An Efficient Algorithm for Mining Association Rules in Large Databases. VLDB'95: Proceedings of the 21th International Conference on Very Large Databases.

Schuster A. and Wolff R. and Trock D. (2003). A High-Performance Distributed Algorithm for Mining Association Rules. ICDM'03: Proceedings of the Third IEEE International Conference on Data Mining.

Soo Park J. and Ming-Syan Chen and Philip S. Yu (1995). An effective hash-based algorithm for mining association rules. SIGMOD'95: Proceedings of the 1995 ACM SIGMOD international conference on Management of Data.

Tibshirani R. and Walther G. and Hastie T. (2000). Estimating the number of clusters in a dataset via the Gap statistic. Technical report, Stanford University, March 2000.

Xu R. and Wunsch D. (2005). Survey of Clustering Algorithms. IEEE Transactions on Neural Networks, Vol. 16, May 2005.

Xu X. and Jager J. and Kriegel H.-P. (1999). A Fast Parallel Clustering Algorithm for Large Spatial Databases. Journal of Data Mining and Knowledge Discovery, Vol. 3, 1999.

Zhang B. and Hsu M. and Dayal U. (1999). K-Harmonic Means - A Data Clustering Algorithm. Technical report, HP Labs, 1999.

Zhang B. and Forman G. (2000). Distributed Data Clustering Can be Efficient and Exact. Technical report, HP Labs, 2000.

Zobel J., Moffat A. (2006). Inverted Files for Text Search Engines, Journal of ACM Computing Surveys, Vol. 38, No.2, Article 6.

Data Mining and Knowledge Discovery in Real Life Applications

Edited by Julio Ponce and Adem Karahoca

ISBN 978-3-902613-53-0
Hard cover, 436 pages
Publisher I-Tech Education and Publishing
Published online 01, January, 2009
Published in print edition January, 2009

This book presents four different ways of theoretical and practical advances and applications of data mining in different promising areas like Industrialist, Biological, and Social. Twenty six chapters cover different special topics with proposed novel ideas. Each chapter gives an overview of the subjects and some of the chapters have cases with offered data mining solutions. We hope that this book will be a useful aid in showing a right way for the students, researchers and practitioners in their studies.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Lamine Aouad, An Le-Khac and Tahar Kechadi (2009). Knowledge Discovery on the Grid, Data Mining and Knowledge Discovery in Real Life Applications, Julio Ponce and Adem Karahoca (Ed.), ISBN: 978-3-902613-53-0, InTech, Available from: http://www.intechopen.com/books/data_mining_and_knowledge_discovery_in_real_life_applications/knowledge_discovery_on_the_grid

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen