An Efficient Support Management Tool for Distributed Data Mining Environments

Nhien-An Le-Khac
School of Computer Science and Informatics
University College Dublin
Dublin 4, Ireland
Email: an.lekhac@ucd.ie

Lamine M. Aouad
School of Computer Science and Informatics
University College Dublin
Dublin 4, Ireland
Email: lamine.aouad@ucd.ie

M-Tahar Kechadi
School of Computer Science and Informatics
University College Dublin
Dublin 4, Ireland
Email: tahar.kechadi@ucd.ie

Abstract—Today, a deluge of data is collected from different fields. These massive amounts of data which are often geographically distributed and owned by different organisations are being mined. As consequence, a large mount of knowledge is being produced. This causes the problem of efficient knowledge management in distributed data mining (DDM). The main aim of DDM is to exploit fully the benefit of distributed data analysis while minimising the communication overhead. Existing DDM techniques perform partial analysis of local data at individual sites and then generate global models by aggregating the local results. These two steps are not independent since naive approaches to local analysis may produce incorrect and ambiguous global data models.

To overcome this problem, we present a tool called "knowledge map" to easily and efficiently represent mined knowledge in a large scale distributed platform such as Grid. This will also facilitate the integration/coordination of local mining processes and existing knowledge to increase the accuracy of the final models. This approach is tested on very large datasets and the results are very promising.

I. INTRODUCTION

While massive amounts of data are being collected and stored from not only science fields but also industry and commerce fields, the efficient mining and management of useful information of this data is becoming a scientific challenge and a massive economic need. This led to the development of distributed data mining (DDM) techniques [12][13] to deal with huge and multi-dimensional datasets distributed in large number of sites. This phenomenon leads to the problem of managing the mined results, so called knowledge, which becomes more complex and sophisticated. This is even more critical when the local knowledge of different sites are owned by different organisations. Existing DDM techniques is based on performing partial analysis on local data at individual sites and then generating global models by aggregating these local results. These two steps are not independent since naive approaches to local analysis may produce incorrect and ambiguous global data models. In order to take the advantage of mined knowledge at different locations, DDM should have a view of the knowledge that not only facilitates their integration but also minimises the effect of the local results on the global models. Briefly, an efficient management of distributed knowledge is one of the key factors affecting the outputs of these techniques.

Recently, many research projects on knowledge management in data mining were initiated [23][11][1]. Their goals are to tackle the knowledge management issues as well as present innovative and efficient approaches. However, most of them propose solutions for centralised data mining and only few of them have attempted the issues of large scale distributed data mining. Moreover, some recent research works[4] have just provided a manner of managing knowledge but not the integration and coordination of these results from local results.

In this paper, we propose a "knowledge map", an approach for managing knowledge (mined results) of distributed data mining tasks on large scale distributed systems and also supporting the integration views of related knowledge. The concept of knowledge map has been efficiently exploited in managing and sharing knowledge[19] in centralised systems but not in distributed systems. Our main goal is to provide a simple and efficient way to handle a large amount of knowledge collected in Grid environments. This knowledge map helps to retrieve quickly any results needed with a high accuracy. It will also facilitate the merging and coordination of local results to generate global models. This knowledge map is one of the key layers of ADMIRE[14] (Fig.1), a framework based on Grid platform for developing DDM techniques to deal with very large and distributed heterogeneous datasets in both commercial and academic applications.

The rest of this paper is organised as follows: In section II we give a background of knowledge representation and knowledge map concept as well as related projects. We present
the architecture of our knowledge map in section 3. Section 4 presents knowledge map’s operations. An evaluation of this approach is presented in section 5. Finally, we conclude in Section 6.

II. Background

In this section, we present first some methods for representing knowledge in data mining. Secondly, we discuss the concept of knowledge map and its use in managing the results.

A. Knowledge representation

There are many different ways of representing mined knowledge in literature, such as decision tables, decision trees, classification rules, association rules, instance-based and clusters. Decision table is one of the simplest ways of representing knowledge. The columns contain set of attributes including the decisions and the rows represent the knowledge elements. This structure is simple but it can be sparse because of some unused attributes. Decision tree approach is based on “divide-and-conquer” concept where each node tests a particular attribute and the classification is given at the leaves level. However, it has to deal with missing value problem. Classification rules [8] is a popular alternative to decision tree. This approach uses production rules [2], called cause-effect relationships, to express the knowledge. A decision tree is used to represent the relationship between the rules. Association rules [8] are kind of classification rules except that they can predict any attribute and this gives them the freedom to predict combinations of attributes too. Moreover, association rules are not intended to be used together as a set, as classification rules are. The instance-based knowledge representation uses the instances to represent what is mined rather than inferring a rule set and store it instead. The problem is that they do not make explicit the structures of the knowledge. In the cluster approach, the knowledge can take the form of a diagram to show how the instances fall into clusters. There are many kinds of cluster representations such as space partitioning, Venn diagram, table, tree, etc. Clustering [8] is often followed by a stage in which a decision tree or rule set is inferred that allocates each instance to its cluster. Other knowledge representation approaches, such as Petri net [21], Fuzzy Petri nets [5] and G-net [7] were also developed and used.

B. Knowledge map concept

A knowledge map is generally a representation of the knowledge or results of the data mining techniques [6][9][24]. It basically helps to detect the sources of knowledge and their structures by representing the elements and structural links of the application domains. Some kind of knowledge map structure that can be found in literature are: hierarchical/radial knowledge map, networked knowledge map, knowledge source, map and knowledge flow map.

Hierarchical knowledge map, so-called concept map [19], provides one model for the hierarchical organization of knowledge: top-level concepts are abstractions with few characteristics. Concepts of the level below have detailed traits of the super concept. The link between concepts can represent any type of relations as “is part of”, “influences”, “can determine”, etc. A similar approach is radial knowledge map or mind map [3], which consists of concepts that are linked through propositions. However, it is radially organised. Networked knowledge map is also called causal map which is defined as a technique "for linking strategic thinking and acting, making sense of complex problems, and communicating with others what might be done about them" [3]. This approach is normally used for systematizing knowledge about causes and effects. Knowledge source map [9] is a kind of organisational charts that does not describe functions, responsibility and hierarchy, but expertise. It helps experts in a specific knowledge domain. The knowledge flow map [9] represents the order in which knowledge resources should be used rather than a map of knowledge.

C. Related works

Little research work on knowledge map is given in [10][17]. However, these few projects were not in the context of DDM. In the context of DDM on Grid platform, the Knowledge Grid project [4] proposed an approach to manage it by using Knowledge Discovery Service. This module is responsible for handling meta-data of not only knowledge obtained from mining tasks but also all kinds of resources such as hosts, data repositories, used tools and algorithms, etc. All metadata information is stored in a Knowledge Metadata Repository. However, this approach does not provide a management of knowledge metadata in their relationships to support the integration view of knowledge as well as the coordination of local the mining process. There is moreover no distinct separation in between resource, data, and knowledge.

Until now, to the best of our knowledge, in spite of a popularity gain of DDM applications, there is only our system [15] that provides knowledge map layer for DDM applications on a grid type platforms. This constitutes one of the motivations of our research to provide a fully integrated view of knowledge and to facilitate the coordination of local mining processes to increase the accuracy of the final models.

III. Architecture of Knowledge Map

The knowledge map (KM) does not attempt to systematize the knowledge itself but rather to codify "knowledge about
knowledge”. In our context, it facilitates DDM by supporting users coordination and interpretation of the results. The objectives of our KM architecture are: provide an efficient way to handle a large amount of data collected and stored in large scale distributed system; retrieve easily, quickly, and accurately the knowledge; and support the integration process of the knowledge. We propose an architecture of the KM system as shown in Fig.2, 3 and 4 to achieve these goals. KM consists of the following components: knowledge navigator, knowledge map core, knowledge retrieval, local knowledge map and knowledge map manager (Fig.2). From now on, we use the term “mined knowledge” to represent for knowledge built from applications.

A. Knowledge navigator

Usually, users may not exactly know the mined knowledge they are looking for. Thus, 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 metadata, called meta-knowledge, which includes related information such as data mining task used, data type, and a brief description of this knowledge and its location. For example, a user may 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 knowledge 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. It will interact with knowledge retrieval component to collect all mined knowledge from chosen locations. In a distributed system, knowledge navigator component is implemented in each site.

B. Knowledge map core

This component (Fig.3) is composed of two main parts: concept tree repository and meta-knowledge repository. The former is a knowledge-base 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 [19]. A node of this tree, so called concept node that represents a sub-application domain and each concept node includes a unique identity, called concept Id, in the whole concept tree repository and a name of its sub-application domain. The content of each concept tree is defined by the administrator before using the KM system. The concept tree repository could also be updated during the runtime. In our approach, a mined knowledge is assigned to only one sub-application domain and this assignment is given by the users. 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.

Meta-Knowledge repository (Fig.3): this handles metadata of the knowledge from different sites. A knowledge is mapped to a knowledge object and its metadata is represented by a meta-knowledge entry in this repository. As shown in Fig.3 for example, the concept tree repository contains an application domain named “meteorology” which includes sub-application domains such as “weather forecasting”, “storm” and “climate”. And then, “thunder storm”, “tropical cyclone” and “tornado” are parts of “storm”. This figure also shows an example of a meta-knowledge entry in XML format. Based on this information, users could determine which mined knowledge they want to extract.

The goal of KM core, is not only detecting the sources of knowledge but also representing their relationships with concepts of application domain. This component could be implemented in a master site assigned to a group of sites depending on the topology of the system. The creation and maintenance of this component as well as its operations such as knowledge retrieving will be presented in section IV.

C. 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 returns knowledge acquired.

D. Local knowledge map

This component (Fig.4) is located in 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 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. Theses attributes are already explained in the section Knowledge map core above. This meta-knowledge is also submitted to the Knowledge map core and will be used in meta-knowledge entry of its repository for use at the global level. The representative of a knowledge
entry depends on a given mining task. KM supports two kinds of representatives: one for clustering task and another for association rules task. Our system has however the capacity of adding more representative types for other mining tasks.

For the association rule case (Fig.4a), the mined knowledge is represented as a set of the production rules [2]. A rule is the form “IF (left expression) THEN (right expression)” and its attributes are support and confidence [8]. An expression contains a set of items. In order to represent these rules by their items, a representative in our approach consists of two parts: a rule net and an item index table. A rule net is a triple-tuple \( (I, R, A) \) where \( I \) is a set of items, \( R \) is a set of rules and each rule has an unique Id, its attributes and its creating information. \( A \) is a set of edges called rule item relations. \( A \) should satisfy the following: \( A \subseteq (I \times R) \cup (R \times I) \). The Index Table is a data structure that maps items to the rule net. For example, the index of a book maps a set of selected terms to page numbers. There are many different types of index described in literature. In our approach, the index table is based on inverted list [25] technique because it is one of the most efficient index structures [26]. This index table consists of two parts: items and a collection of lists, one list per item, recording the identity of the rule containing that item. For example (Fig.4a), we assume that the term "cloud" exists in rules 25, 171, 360, so its list is \{25, 171, 360\}. This index table also expresses the relationship between items and their corresponding rule. By using this table, rules which are related to the given items will be retrieved by the intersection of their lists, e.g. the list of term "pressure" is 20, 171 so the ID of rule that contains "cloud" and "pressure" is 171. This ID is then used to retrieve the rule and its attributes. Besides, a rule can be created by using one or more rules, so its creating information keeps this link (Fig.4c).

For the clustering case (Fig.4b), a representative of a 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, which it is also defined by the user, depends on the clustering algorithm used. The metadata of these fields is also included in the representative. KM allows the user to define this metadata with both scalar and vector data type. A cluster also contains information about its creating. 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). Cluster identity is unique in its knowledge entry. For example, a knowledge entry which is created by a variance-based clustering algorithm [16] on test datasets, has its representative in XML format as shown in Fig.5. Clusters were created in this example and Fig.4c shows an example of integrating link. Note that in Fig.4c, representative (ii) and (iii) belong to the same knowledge.

E. Knowledge map manager

Knowledge map manager is responsible for managing and coordinating the local knowledge map and the knowledge map core. For local knowledge map, this component provides primitives to create, add, delete, update knowledge entries and their related components (e.g. rule net and item index table) in knowledge repository. It also allows to submit local meta knowledge to the 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. A key role of this component is to keep the coherence between the local knowledge map and the knowledge map core.
IV. KNOWLEDGE MAP OPERATIONS

In this section, we present basic operations of knowledge map system including adding, retrieving, and maintaining the KM system. We first create the concept tree repository with some predefined concept trees. The administrator can then update its content.

A. Adding new knowledge

For any new mined knowledge, its corresponding metadata and its representative are generated and mapped to a knowledge object. This object will be added to the local meta-knowledge repository with an appropriate concept Id. This meta-knowledge is then submitted to the core meta-knowledge repository. The Adding operation is realized via the primitive "put".

B. Update/delete knowledge

KM allows users to update or to delete an existing knowledge metadata via "update" and "delete" primitives. These operations are executed at local site and then the system will automatically update knowledge map core to ensure the coherence between core and local knowledge map. This operation is moreover atomic.

C. Knowledge searching/retrieving

These operations are functions of find/retrieve primitives. KM supports different levels of search: concepts or metadata of mined knowledge. At the concept level, KM allows the user to search and retrieve concepts acquired through their identity or name. The search operation can be done using different criteria such as: concept (e.g. search all meta knowledge of a selected concept), mining task and algorithm used to build its knowledge. The retrieve operation is performed through the knowledge identity and the location of the knowledge needed. This process returns a knowledge object. This operation is executed both locally and globally, i.e. users can retrieve the knowledge needed at its local site or from a group of sites of the system.

V. IMPLEMENTATION AND EXPLOITATION

KM is implemented in Java and its current version uses Java RMI as the communication middleware. Moreover, it can easily use other communication middleware models. In this version, repositories of KM core and Local KM are XML documents. One site of the system is chosen as a host to store the meta-knowledge repository. The KM runtime includes a set of KM Daemon (Fig.6). Each local site has one KM Daemon that is responsible for processing of local/remote requests. These KM Daemons are created at the start by using the primitive "init". The primitive "stop" will terminate all the KM Daemons. A KM application can send request to one or many remote sites. As shown in (Fig.6), for example, first find all the meta-knowledge needed via primitive "find" (Fig.6a). This action is composed of four steps: a request is sent to the host (1) to look for the meta-knowledge needed. Then, this will be retrieved (2) and sent back to the source site (3), and it extracts the results as meta-knowledge objects (4).

The application extracts knowledge via primitive "retrieve" (Fig.6b). This action is also composed of four steps: (1) requests are sent to the appropriate sites; (2) retrieve the knowledge found at each site; (3) send back to the source site via KM Daemon; (4) extracts results as knowledge objects.

Next, we present two scenarios of using knowledge map. In the first one, mined knowledge already exist at different sites in the system. In the second scenario, a distributed data mining task is executed on a system such as cluster or grid. In the first scenario, if the meta-knowledge of those mined knowledge have not been handled by the knowledge map, then the first step is to use knowledge map tool to create knowledge objects and store it in each local KM. Their meta-knowledge will be automatically submitted to the meta-knowledge repository at the knowledge core map. The users can also add more appropriate concepts to their knowledge. This step needs an interaction with the user who created these knowledge. The user, then, can exploit these meta-knowledge and knowledge object in their integration process or only explore the knowledge. In the second scenario, after the local mining process has been finished, local mined knowledge is built in each site. Its meta-knowledge is created and is stored in appropriate repository. Then, the integration process uses these meta-knowledge to retrieve information required.

VI. EVALUATION

We are using this tool in our framework [15][16]. It is difficult to evaluate our approach by comparing it to other systems because it is unique so far. Therefore, we evaluate this new approach 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 node 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 the B-tree model. However, the number of concept entries as well as of concept nodes of a concept tree is smaller 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 $O(\log M)$. The complexity of retrieving a knowledge object depends on the number of knowledge entries $m$ in local KM. Therefore, this complexity is $O(\log M + C \log m)$, where $C$ is the communication cost.

Next, we estimate the knowledge map architecture. Firstly, the structure of concept tree is based on the concept map [19], which is one of the advantages of this model. We can avoid the problem of semantic ambiguity as well as reduce the domain search to improve the speed and accuracy of the results. 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 site (or server), which is not feasible in very large distributed system such as Grid. By representing knowledge metadata in their relationship links, the goal is to provide an integration view of these knowledge. Furthermore, the use of rule net and index table for representing the rules mined from the task datasets create the local knowledge map to be moreover a map of knowledge elements by representing relationships between items and rules.

Finally, our new 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 as [20] or applying compressed technique as discussed in [27]. 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.

**VII. Conclusion**

In this paper, we presented an architecture of the knowledge map layer. This innovative tool aims at managing effectively the mined knowledge in large scale distributed platforms. The purpose of this research is to provide a knowledge map to facilitate the visualization of the results as well as to provide a viable environment for the DDM applications. Throughout estimations of each component and it function, we can conclude that knowledge map is an efficient and flexible system in a large and distributed environment. It satisfies the needs for managing, exploring, and retrieving the mined knowledge of DDM in large distributed environment.

This knowledge map is integrated in the ADMIRE framework. Experimental results on real-world applications are also produced [16] and this will allow us to test and evaluate deeply the system robustness and the distributed data mining approaches at very large scale.

**REFERENCES**


