Systems and techniques for distributed and stream data mining

Domenico Talia, Paolo Trunfio
{talia|trunfio}@deis.unical.it
Università della Calabria, Rende (CS), Italy

Salvatore Orlando, Raffaele Perego, Claudio Silvestri
{orlando|silvestri}@dsi.unive.it
raffaele.perego@isti.cnr.it
ISTI-CNR, Pisa, Italy

CoreGRID Technical Report
Number TR-0045
July 4, 2006

Institute on Knowledge and Data Management

CoreGRID - Network of Excellence
URL: http://www.coregrid.net

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

Systems and techniques for distributed and stream data mining

Domenico Talia, Paolo Trunfio
{talia|trunfio}@deis.unical.it
Università della Calabria, Rende (CS), Italy

Salvatore Orlando, Raffaele Perego, Claudio Silvestri
{orlando|silvestri}@dsi.unive.it
raffaele.perego@isti.cnr.it
ISTI-CNR, Pisa, Italy

CoreGRID TR-0045
July 4, 2006

Abstract

Nowadays huge amounts of electronic data are naturally collected in distributed sites, due to either plural ownership or geographical distribution of the processes that produce data. Moving data to a location for extracting useful and actionable knowledge is usually considered unfeasible, for either policy or technical reasons. It thus becomes mandatory to mine them by exploiting the multiple distributed resources close to data repositories. This requires to develop novel distributed data mining (DM) algorithms and systems, able to return a global knowledge by aggregating multiple local results. Besides the distributed nature of data under analysis, nowadays researchers have also to consider the streaming nature of them, which entails approximate on-line methods to mine data stream. This report introduces requirements, design issues, and typical solutions of distributed DM algorithms. We survey some of the most important works that recently appeared in the literature on this topic, included the latest proposals regarding P2P, highly distributed, approximate algorithms. Moreover, we discuss some of the most important proposals concerning stream DM algorithms. Such DM algorithms can be considered as the building blocks for realizing seamless distributed knowledge extraction processes and systems. Such systems can take advantage of the recent advances in computational and data Grid, which many researchers consider as the enabling technology for also developing high-performance knowledge discovery processes and systems. In this report we analyze some significative examples of distributed and Grid-oriented knowledge discovery systems.

1 Introduction

Data Mining (DM) can be defined as the automatic extraction of useful and actionable knowledge, hidden in huge amounts of data. It is usually considered as the computational intensive phase of a more complex process of knowledge discovery, which also involves other important phases, like the cleaning of input raw data or the presentation of resulting knowledge.

The design of a system for knowledge discovery, and in particular the algorithms and tools for DM, have to consider that, nowadays, input data are naturally distributed, due to either a plural ownership, or a geographical distribution of the processes which produce data. Moving all such data to one single location for processing could be impossible due to either policy or technical reasons. Furthermore, communications between entities owning parts of data may be not particularly fast and immediate. In this context, the communication efficiency of an algorithm is often more important than the accuracy of its results.

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

Typically, distributed data mining (DDM) algorithms thus involve local data analysis, followed by the generation of a global data model through the aggregation of the local results.

As suggested before, there are several cases in which data can be distributed among different sites/nodes. In case of cellular phone networks, each group of cells may have its separate database for performance and resilience reasons. At the same time, other information about which customers own a given device is only available at the account department, and is kept separate for privacy reasons. Where a particular data can be found influences the kind of solutions a problem can have. So, it becomes important to specify in which context an algorithm can profitably be used.

Moreover, in distributed data mining, it is also important to distinguish between different network speeds, and their consequences over the chosen solution in terms of computation/communication cost. In other words, considering both costs of communication and computation, and the result accuracy requirements, we can trade-off between a pure distributed approach based on moving models, and a more traditional one, based on moving data toward a centralized mining service. Privacy concerns, on the other hand, may prevent the exploitation of a moving data approach, unless original data have been perturbed to avoid disclosing private information.

Nowadays the data distribution requirements must often be considered in conjunction with the streaming nature of data sources. Many critical data mining applications require a nearly immediate result based on single (multiple) continuous stream (streams) of data. The infinite nature of these data sources is a serious obstacle to the use of most of the traditional methods, since available computing resources are limited. Therefore, in the last couple of years, on-line mining of data streams for knowledge discovery has thus become a very important research direction.

In the following we discuss in more detail some of the general issues and requirements related to the above topics, and illustrate the organization of this report.

Homogeneous and heterogeneous data distribution. In order to discuss the two major classes of data distribution, we can still consider the cellular phone domain. An example of *homogeneous data distribution* corresponds to the case in which each node owns its database, containing the portions of logs referring to the activities of a device, in the area controlled by a specific group of antennas. Every local database contains different data, but the kind of information is exactly the same for every node. On the other hand, if we are also interested in data about each customer, nodes owning different kinds of data need to cooperate. In the example, the cell database could contain the information that a device stopped for several hours in the same place, whereas the accounting department database knows which customer is associated with that device and its home address. This situation is referred to as *heterogeneous data distribution*.

Communication bandwidth and latency issues. A key factor in the implementation of distributed algorithms is the kind of communication infrastructure available. An algorithm suitable for nodes connected by a high speed network can be of little use if a high latency/low bandwidth WAN is available. Furthermore, for algorithms which entail several blocking communications/synchronization, a high latency is definitely a serious issue. Hence, efficient distributed algorithms need to exchange a few data, and avoid global synchronization as much as possible, also because computation can be inherently unbalanced due to node or data heterogeneity.

Peer-to-Peer Mining. A new interesting example of distributed mining is concerned with peer-to-peer (P2P) highly distributed databases. The P2P systems, besides file sharing networks, also include, to some extent, GRID computing environments and sensor networks. Large-scale P2P mining may be very different from the more traditional DDM. For example, in such systems, there can not be any global synchronization. Nodes must act independently of each other, on the basis of information exchanged with only a limited neighborhood. Their progress in knowledge extraction thus becomes speculative, and the global system arrives at convergence with some delay and approximation in the final results.

Privacy and Multi-party Computation. Distributed data mining could occur between mutually un-trusted parties, or even between competitors. For example, several competing financial organizations might be interested in jointly derive a predictive model from data owned by their organizations. If nobody can be trusted enough to collect all the input data before running a mining tool, privacy will become a primary concern. This problem is referred to as Secure Multi-party Computation Problem (SMC) in the literature.

DDM algorithms, since are mainly based on merging local models extracted from distributed sites, fall in this class, since models generated locally might still contain sensitive information that a partner does not want to disclose.

Stream mining. The streaming nature of sources entails the need of processing data as they arrive. The amount of previously happened events is usually overwhelming, so they can be either dropped after processing or archived separately in secondary storage. In the first case access to past data is obviously impossible, whereas in the second case the cost for data retrieval is likely to be acceptable only for some ”ad hoc” queries, especially when several scan of past data are needed to obtain just one single result.

Other important differences with respect to having all raw data available and processed at once regard the results. As previously explained, both data and results evolve continuously. Hence a result is referred to a part of the stream. Obviously, an algorithm suitable for stream data should be able to compute the ”next step” solution starting from the previously known one and current data.

Two models can be used for mining streams. The first one is the *landmark model*, according to which patterns mined from data streams are measured from the start of the stream up to the current moment. On the other hand, if the patterns extracted are time-sensitive, a user can be more interested in theirs changes and trends, so that a *sliding window model* can be more suitable. An alternative model that can be exploited to mine streams is the *tilted-time* window one, which is based on the fact that people are often interested in recent changes at a fine granularity, and in long term changes at a coarse granularity.

Systems to support knowledge discovery processes. In order to support a complete knowledge discovery process over emerging highly distributed platforms like computational Grids, the most important issue is the integration of two main requirements: synthesizing useful and usable knowledge from data, and performing complex large-scale computations leveraging the Grid infrastructure. Moreover, since current research activity on the next generation Grid and Web is focusing on designing and implementing its mechanisms following the *Service Oriented Architecture* (*SOA*) model, we have to choose a SOA-based technology to develop such system.

Organization of the report

In Section 2, we survey some of the most recent proposals concerning DDM and P2P data mining (Section 2.1), and also discuss some on-going research activities concerning stream mining problems, such as online aggregate evaluation and approximation, frequency counting, classification and clustering (Section 3). Section 4 surveys some significative examples of distributed and Grid-oriented knowledge discovery systems. Finally, Section 5 draws some conclusions and future work.

2 Distributed and stream mining

The use of distributed systems is continuously spreading in several applications domains, where data are naturally distributed among several parties/entities, and often evolve continuously. Extracting valuable knowledge from raw data produced by distributed parties, in order to produce a unified global model, may present various challenges related to either the huge amount of managed data, or their physical location and ownership.

The centralized model, based on uploading data towards a centralized point where a high performance algorithm is run, is not suitable for such distributed environments. We can recognize numerous reasons for this. The main ones regard the possible long response time of such mining process, also due to the lack of proper use of existing distributed resources, and the inappropriateness of centralization due to constraints like limits on the network bandwidth available, or on the memory space available in the centralized server [34].

We thus need data mining algorithms and systems that pay particular attention to the distributed features of data, computation and communication resources. It is necessary to devise new scalable approaches for distributed processing of data, controlled by resource available, features of processed data, and also human factors.

The widespread use of wireless networks, and the availability of ubiquitous sensors collecting data for disparate purposes [5], open new problems for DDM, calling for algorithms that reduce power-consuming communications as much as possible in favor of local processing of data. Moreover data can arrive in streams [3], and this may call for algorithms that guarantee approximate results.

2.1 Distributed and P2P mining algorithms

DDM algorithms typically work by producing a local model per site (see Figure 1). Unfortunately, even if local models are coherent and accurate with respect to local site repository, inferring a global model by aggregating the local models may be very complex. A possible solution is to centralize a portion of local data on the node that will be responsible for model aggregations. Alternatively, we can compute in an exact way part of the global model. This can be used to compensate for the lacking of some global knowledge if we aggregate by only starting from local models.

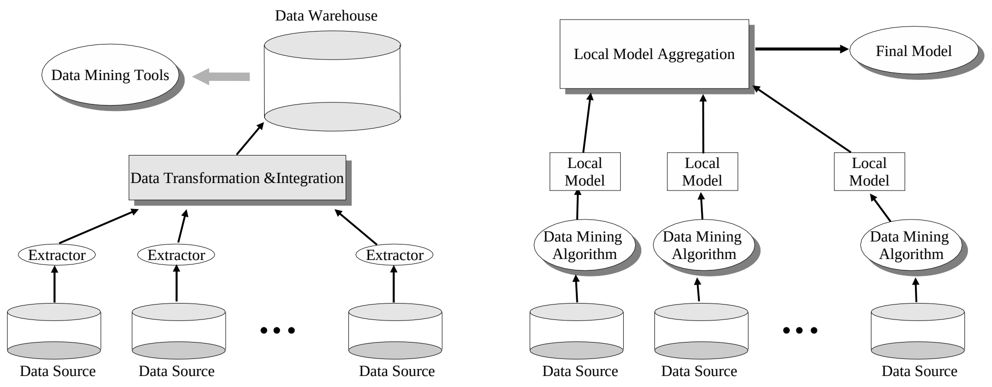

Figure 1: The centralized data-warehouse approach vs. a typical distributed data mining framework (from [34]).

Distributed classifiers Distributed classifiers are the best-known examples of DDM algorithms. Such algorithms are rooted in the well-known ensemble approach [16, 33], originally exploited to improve accuracy of predictive models. In practice, multiple models are extracted by homogeneous data partitions, also using different methods. The local models are aggregated by adopting a voting (weighted or un-weighted) schema. This ensemble approach can be straightforwardly realized in a distributed environment. A notable example of the ensemble approach is the meta-learning distributed framework, introduced by Chan and Stolfo [35]. In this framework the local base models, collected at central site, are used to generate meta-data, i.e. a set of validation data classified on the basis of the predictions generated by the local classifier. Finally, the global model is generated from all such meta-data, thus producing the final meta-classifier. However, the option of producing the global classifier by counting votes returned by base classifiers. A problem with the ensemble meta-learned approach lies in the nature of data repositories. For heterogeneous data, Park et al. noted that it may become impossible to extract inter-site patterns by only merging base local classifiers. In order to solve this problem, part of data needs to be centralized. An innovative approach to mine distributed heterogeneous data has been proposed by Kargupta et al. in their Collective Data Mining framework [9].

Distributed Frequent Pattern Mining. A number of parallel algorithms for solving the FIM have been proposed in the last years [2, 22]. Most of them can be considered parallelizations of the well-known Apriori algorithm.

Zaki authored a good survey on ARM algorithms and relative parallelization schemas [40]. Agrawal et al. [2] proposed a broad taxonomy of the parallelization strategies that can be adopted for Apriori on distributed-memory architectures. The described approaches constitute a wide spectrum of tradeoffs between computation, communication, memory usage, synchronization, and the use of problem-specific information. The Count Distribution (CD) approach follows a data-parallel according to which the transaction database is statically partitioned among the processing nodes, while the candidate set C_k is replicated. At each iteration every node counts the occurrences of candidate itemsets within the local database partition. At the end of the counting phase, the replicated counters are aggregated, and every node builds the same set of frequent itemsets F_k. On the basis of the global knowledge of F_k, candidate set C_{k+1} for the next iteration is then built. Inter-Node communication is minimized at the price of carrying out redundant duplicate computations in parallel. Count Distribution is an algorithm that can be realized in a distributed setting, since it based on a partitioned dataset, and also because the amount of information exchanged between nodes is limited. The other two methods proposed by Agrawal et al., Data and Candidate distribution, require moving the dataset.

Unfortunately in a distributed environment such dataset is usually already partitioned and distributed on distinct sites, and can not be moved for several reasons, for example due to the low latency/bandwith network that connects the sites.

Several DDM FIM algorithms have been proposed, aimed at reducing the amount of communications involved in the Count Distribution method. FDM [10] constitutes an attempt to reduce the amount of communication entailed in the sum-reduction of the local counters in the CD parallelization of the Apriori algorithm. In particular, the counters that are sum-reduced are only the ones that correspond to candidates that have been identified as frequent by at least one party. Schuster and Wolff [36] then introduced DDM, whose aim is to reduce the number of messages exchanged by FDM, since this number in presence of non-homogeneity in the database partitions quickly becomes similar to the ones exchanged by CD. The basic idea of DDM is to verify that an itemset is frequent before collecting its support from every party. This negotiation before the actual reduction must, of course, be not expensive. The heuristics adopted allows those parties that have more convincing evidence that a given itemset is frequent, to start the negotiation. This occurs, for example, if a party computes, by itself, a support of an itemset that is already larger than the global support threshold. The same authors extend the idea of DDM to a dynamic large scale P2P environment, i.e., a system based on utilizing free computational/storage resources on non-dedicated machines, where nodes can suddenly depart/join along with the associated database, thus modifying the global result of the computation. Schuster and Wolff noted that previous proposals of distributed FIM use broadcasts, global synchronization, various level of centralization, none of which can be managed in large-scale distributed systems. Moreover, no ones acknowledge the possibility of node failures. They thus contribute the reduction of the FIM problem in this highly distributed setting to the well-studied problem of distributed majority votes [38]. The P2P algorithms thus combines sequential miners, executed locally at each node, with a majority voting protocol to discover, at each node, all of the association rules that exist in the combined database. Note that also in DDM the distributed ARM algorithm has been seen as a decision problem in which the participating nodes must decide whether each itemset is frequent or not. The difference here is that the consensus is not reached by using broadcast and global synchronization, but only communications between neighboring nodes.

Distributed Clustering. Most of the distributed clustering algorithms can straightforwardly be derived from their parallel counterparts. Since these parallel algorithms are based on the assumptions that a single dataset is partitioned, in order to adapt them for a distributed setting, the databases owned by each distributed site must be homogeneous.

In [34] some center-based clustering algorithms are surveyed. One of the most notable example is the work of Forman and Zhang [18], in which they extend a class of parallel iterative clustering algorithms (K-Means, K-Harmonic Means, and EM) to inherently distributed databases. All the algorithms are based on the global reduction of so-called sufficient statistics, followed by a broadcast of the final result. A popular statistics concerning the goodness of a clustering is, for example, the sum of the mean square error (MSE) of each data point from its own cluster center.

In [39] Xu, et al. discuss a parallel version of a very popular data mining clustering algorithm, the density-based DBSCAN one. The target platform is a shared-nothing cluster of computers, which can be thought as a distributed system with a more efficient interconnection network. The algorithm is based on a $dR^* - tree$, a distributed spatial index structure. Every computer has its own local data, while the replicated index provides an efficient access to data. The key idea is the exploitation of a master which is responsible for merging candidate clusters found by distinct slaves.

Finally, in a recent work [14], besides presenting an overview of some data mining algorithms for peer-to-peer environments, the authors also describe both exact and approximate decentralized algorithms for the problem of monitoring and computing clusters in data residing at different nodes of a P2P network.

3 Stream mining algorithms

In recent applications data are modeled best not as persistent relations, but rather as transient data streams. Examples of such applications include financial applications, network monitoring, security, telecommunications data management, web applications, manufacturing, sensor networks, moving object trajectories and RFID tracing applications, and many others.

The management of multiple and distributed streams, produced at different rates, arriving in an unpredictable and unbounded way, appears to yield some fundamentally new research problems. Since in general such streams may arrive in burst and are unbounded, due to real-time and memory constraints we cannot store them on a persistent memory, and it becomes mandatory to examine each incoming record once or a limited number of time. Moreover, sampling is often the only option to face the above constraints.

Since approximate query answers often suffice, and this is particularly true in trend/pattern analyses, typical application have to maintain in real time small synopses of the data streams. Such synopses must be built in order to provide good-quality approximate answers.

Streams are naturally ordered, and tuples can be associated with logical or physical timestamps. Windows covering part of timestamped stream tuples permit for only considering portions of stream data. For example, *sliding windows* prevent stale data from influencing analysis and statistics. Such windows are also useful for approximation in face of bounded memory. A useful way to discount for past tuples of the stream is also the *fading window* model. For example, by introducing a fading factor ϕ, we can reduce the influence on a given statistics, e.g., the support count of an itemset by reducing the absolute support supplied to the itemset by an old block of past transactions.

A good work that surveys a lot of past studies on stream data management and stream (continuous) query processing is [3]. Stream data mining is a relatively new research topic. The main focus of some important papers has been on stream data classification (such as [17, 24, 29]) and stream clustering (such as [32, 21]).

In [17] Domingos and Hulten discuss VFDT (Very Fast Decision Tree learner), a new stream algorithm to build a decision tree. It is able to build a tree using constant memory and constant time per example. It uses Hoeffding bounds to select enough stream examples for choosing the right test attribute for each internal node of the tree. at any given node. Indeed, only the first examples that arrive from the stream need are actually used to choose the split attribute at the root. The following examples are passed through the induced portion of the tree, until they reach a leaf. If needed, they will be used to select the right split attribute there, and so on recursively.

The classification system tries to learn a discrete function given a set of input and output examples. Unfortunately VFDT makes the assumption that training data is a sample of the examples taken from a stationary distribution. If this assumption is not valid, due two the change in the input distribution, we should observe a change in the target function over time. This problem is known as concept drift. In [24], Hulten, Laurie and Domingos introduce a new stream decisione tree learner, called CVFDT, which is able to cope with concept drifts. CVFDT grows an alternative subtree each time the old one seems to be out-of-date, and replaces the old one when the new one becomes more accurate.

Law and Zaniolo [21] argue that the approach followed by VFDT and CVFDT is interesting from the theoretical point of view, but it can be dif- ficult to apply in real stream environments, characterized by fast input rates and unending data sets, because the significant time required for updating the decision tree. Moreover, due to the statistical bounds, a large amount of samples is needed to build a classifier with reasonable accuracy. Their approach is different, because they propose to build a lazy classifier from the stream, in particular a nearest neighbor (NN) one. While, in general, this approach could be in principle able to solve the problem of the high arrival rate of the stream data, since no explicit classifier must be learnt, it is important to prepare a suitable data structure for accelerating the nearest neighbor lookup. Rather than using binary trees, which may become seriously unbalanced when new stream data arrive, they propose an approach to NN classifiers that only provides approximate answers with error bound guarantees.

Guha, *et al.* [21] focus their attention on clustering data stream. In particular, they deal with a particular definition of clustering, the k-Median objective, whose goal is to identify k centers so that the sum of distances from each point to its nearest center is minimized. The algorithm is based on a facility location algorithm, which might produce more than k centers. The facility location algorithm is however modified to produce exactly k clusters, thus solving the k-Median problem. Finally, the stream algorithm exploits a simple divide-and-conquer core that achieves a constant-factor approximation in small space. This is very important because a stream algorithm must be not only single pass, but has also to use a limited amount of main memory.

The main algorithms [30, 25] to mine frequent itemsets from stream data mining are derived from those for discovering frequent single items. Under particular circumstances, also the apparently simple job of extracting single items may be a highly memory intensive problem [8]. Several relaxed versions of this problem exist, some interesting ones were introduced in [8, 15, 28]. The techniques used for solving this family of problems can be classified into two large categories: count-based techniques [31, 15, 28, 30] and sketch-based techniques [30, 8, 11, 12]. The first ones monitor a limited set of potentially ”interesting” items, using a counter for each one of them. In this case an error arises when an item is erroneously removed from the set, or inserted too late. The second family provides a frequency estimation for every item by using a hash indexed vector of counters. In this case the risk of completely missing the occurrences of an item is avoided, at the cost of looser guarantees on the computed frequencies.

4 Grid platforms for building knowledge discovery systems

Whereas some high-performance parallel and distributed data mining systems have been proposed [27] - see also [6] - there are few research projects attempting to implement and/or support knowledge discovery processes over computational Grids. A main issue here is the integration of two main requirements: synthesizing useful and usable knowledge from data, and performing complex large-scale computations leveraging the Grid infrastructure. Such integration must pass through a clear representation of the knowledge base used in order to translate moderately abstract domain-specific queries into computations and data analysis operations able to answer such queries by operating on the underlying systems [4].

Among them, the Knowledge Grid [7] is a framework for implementing knowledge discovery tasks in a wide range of high performance distributed applications. The Knowledge Grid offers to users high-level abstractions and a set of services by which is possible to integrate Grid resources to support all the phases of the knowledge discovery process, as well as basic, related tasks like data management, data mining, and knowledge representation. Current research activity on the Knowledge Grid is focused on designing and implementing its mechanisms following the *Service Oriented Architecture* (*SOA*) model.

The following Section 4.1 shortly reviews the most significant Grid-oriented knowledge discovery systems.

4.1 Knowledge discovery on Grids

Berman [4], Johnston [26], and some of us [6] claimed that the development of knowledge Grids on top of computational Grids is the enabling condition for developing high-performance knowledge discovery processes and meeting the challenges posed by the increasing demand of power and abstractness coming from complex Problem-Solving Environments. The design of knowledge Grids can benefit from the layered Grid architecture, with lower levels providing middleware support for higher level application-specific services.

In this section we review the most significant systems supporting knowledge discovery processes over distributed/Grid infrastructures. The systems discussed here provide different approaches in running knowledge discovery processes on Grids. We discuss them starting from general frameworks, such as the TeraGrid project, then outlining data-intensive oriented systems, such as DataCutter and InfoGrid, and, finally, describing KDD systems such as Discovery Net and GridMiner, and some significant data mining testbed experiences.

The TeraGrid project is building a powerful Grid infrastructure, called *Distributed TeraScale Facility* (*DTF*), connecting four main sites in USA (the San Diego Supercomputer Center, the National Center for Supercomputing Applications, Caltech and Argonne National Laboratory). Recently, the NSF funded the integration into the DTF of the *TeraScale Computing System* (*TCS-1*) at the Pittsburgh Supercomputer Center; the resulting Grid environment will provide, besides tera-scale data storage, 21 TFLOPS of computational capacity [1]. Furthermore, the TeraGrid network connections, whose bandwidth is in the order of tenths of Gbps, have been designed in such a way that all resources appear as a single physical site. The connections have also been optimized in order to support peak requirements rather than an average load, as it is natural in Grid environments. The TeraGrid adopts Grid software technologies and, from this point of view, appears as a *virtual system* in which each resource describes its own capabilities and behavior through *Service Specifications*. The basic software components are called *Grid Services*, and are organized onto three distinct layers. The *Basic* layer comprises authentication, resource allocation, data access and resource information services; the *Core* layer comprises services such as advanced data management, single job scheduling and monitoring; the *Advanced* layer comprises superschedulers, resource discovery services, repositories etc. Finally, *TeraGrid Application Services* are built using Grid Services. The definition of such services is still under discussion, but they should comprise the support of on-demand/interactive applications, the support of GridFTP interface to data services, etc.

The most challenging application on the TeraGrid will be the synthesis of knowledge from very large scientific data sets. The development of knowledge synthesis tools and services will enable the TeraGrid to operate as a knowledge Grid. A first application is the establishment of the Biomedical Informatics Research Network to allow brain researchers at geographically distributed advanced imaging centers to share data acquired from different subjects and using different techniques. Such applications make a full use of a distributed data Grid with hundreds of terabytes of data online, enabling the TeraGrid to be used as a knowledge Grid in the biomedical domain. The use of the Knowledge Grid services can be potentially effective in these applications.

InfoGrid is a service-based data integration middleware engine designed to operate on Grids. Its main objective is to provide information access and querying services to knowledge discovery applications [20]. The information

[1] http://www.teragrid.org

integration approach of InfoGrid is not based on the classical idea of providing a *universal* query system: instead of abstracting everything for users, it gives a personalized view of the resources for each particular application domain. The assumption here is that users have enough knowledge and expertise to handle the absence of *transparency*. In InfoGrid the main entity is the *Wrapper*; wrappers are distributed on a Grid and each node publishes a directory of the wrappers it owns. A wrapper can wrap information sources and programs, or can be built by composing other wrappers (*Composite Wrapper*). Each wrapper provides: (*i*) a set of query construction interfaces, that can be used to query the underlying information sources in their native language; (*ii*) a set of administration interfaces, that can be used to configure its properties (access metadata, linkage metadata, configuration files). In summary, InfoGrid puts the emphasis on delivering metadata describing resources and providing an extensible framework for composing queries.

DataCutter is another middleware infrastructure that aims at providing specific services for the support of multi-dimensional range querying, data aggregation and user-defined filtering over large scientific data sets in shared distributed environments [2]. DataCutter has been developed in the context of the *Chaos* project at the University of Maryland; it uses and extends features of the *Active Data Repository* (*ADR*), that is a set of tools for the optimization of storage, retrieval, and processing of very large multi-dimensional data sets. In ADR, data processing is performed at the site where data is stored, whereas in Grid environments this is naturally unfeasible, due to inherent data distribution and resource sharing at servers that may lead to inefficiencies. To overcome this issue, in the DataCutter framework an application is decomposed into a set of processes, called *filters*, that are able to perform a rich set of queries and data transformation operations. Filters can execute anywhere but are intended to run on a machine close (in terms of connectivity) to the storage server. DataCutter supports efficient indexing. In order to avoid the construction of a huge single index that would result very costly to use and keep updated, the system adopts a multi-level hierarchical indexing scheme, specifically targeted at the multi-dimensional data model adopted.

Differently from the two environments discussed above, the Datacentric Grid is a system directed at knowledge discovery on Grids designed for mainly dealing with immovable data [37]. The system consists of four kinds of entities. The nodes at which computations happen are called *Data/Compute Servers* (*DCS*). Besides a compute engine and a data repository, each DCS comprises a *metadata tree*, that is a structure for maintaining relationships among raw data sets and models extracted from them. Furthermore, extracted models become new data sets, potentially useful at subsequent steps and/or for other applications.

The *Grid Support Nodes* (*GSNs*) maintain information about the whole Grid. Each GSN contains a directory of DCSs with static and dynamic information about them (e.g., properties and usage), and an execution plan cache containing recent plans along with their achieved performance. Since a computation in the Datacentric Grid is always executed on a single node, execution plans are simple. However, they can start at different places in the model hierarchy because, when they reach a node, they could find or not already computed models. The *User Support Nodes* (*USNs*) carry out execution planning and maintain results. USNs are basically proxies for user interface nodes (called *User Access Points*, *UAPs*). This is because user requests (i.e., task descriptions) and their results can be small in size, so in principle UAPs could be simple devices not always online, and USNs could interact with the Datacentric Grid when users are not connected.

An agent-based data mining framework, called ADaM, has been developed at the University of Alabama [3]. Initially, this framework was adopted for processing large data sets for geophysical phenomena. More recently, it has been ported to the NASA's Information Power Grid (IPG) environment, for the mining of satellite data [23]. In this system, users specify *what* is to be mined (data sets names and locations), *how* and *where* to perform mining (sequence of operations, required parameters and IPG processors to be used). Initially, *thin* agents are associated to the sequence of mining operations; such agents acquire and combine the needed mining operations from repositories that can be public or private, i.e., provided by mining users or private companies. Data is acquired *on-the-fly* in order to minimize storage requirements at the mining site. ADaM comprises a moderately rich set of interoperable operation modules, comprising *data readers* and *writers* for a variety of formats, *preprocessing modules*, e.g., for data subsetting, and *analysis modules* providing data mining algorithms.

The InfoGrid system mentioned before has been designed as an application specific layer for constructing and publishing knowledge discovery services. In particular, it is intended to be used in the Discovery Net system. Discovery Net is a project of the Engineering and Physical Sciences Research Council, at the Imperial College [13] whose main goal is to design, develop and implement an infrastructure to effectively support scientific knowledge discovery processes from high-throughput informatics. In this context, a series of testbeds and demonstrations are being carried out, for using the technology in the areas of life sciences, environmental modeling and geo-hazard prediction.

[2]http://www.cs.umd.edu/projects/hpsl/chaos/ResearchAreas/dc

[3]http://datamining.itsc.uah.edu/adam

The building blocks in Discovery Net are the so-called *Knowledge Discovery Services* (*KDS*), distinguished in *Computation Services* and *Data Services*. The former typically comprise algorithms, e.g., data preparation and data mining, while the latter define relational tables (as queries) and other data sources. Both kinds of services are described (and registered) by means of *Adapters*, providing information such as input and output types, parameters, location and/or platform/operating system constraints, *factories* (objects allowing to retrieve references to services and to download them), keywords and a human-readable description. KDS are used to compose moderately complex data-pipelined processes. The composition may be carried out by means of a GUI which provides access to a library of services. The XML-based language used to describe processes is called *Discovery Process Markup Language* (*DPML*). Each composed process can be deployed and published as a new process. Typically, process descriptions are not bound to specific servers since the actual resources are later resolved by lookup servers (see below).

Discovery Net is based on an open architecture using common protocols and infrastructures such as the Globus Toolkit. Servers are distinguished into (*i*) *Knowledge Servers*, allowing storage and retrieval of knowledge (meant as raw data and knowledge models) and processes; (*ii*) *Resource Discovery Servers*, providing a knowledge base of service definitions and performing resource resolution; (*iii*) *Discovery Meta-Information Servers*, used to store information about the *Knowledge Schema*, i.e., the sets of features of known databases, their types, and how they can be composed with each other.

The GridMiner project at the University of Vienna aims to cover the main aspects of knowledge discovery on Grids. GridMiner is a model based on the OGSA framework [19], and embraces an open architecture in which a set of services are defined for handling data distribution and heterogeneity, supporting different types of analysis strategies, as well as tools and algorithms, and providing for OLAP support. Key components in GridMiner are the Data Access service, the Data Mediation service, and the Data Mining service. Data Access implements the data access to databases and data repositories; Data Mediation provides for a view of distributed data by logically integrating them into virtual data sources (VDS) and allowing to send queries to them and combine and deliver back the results. The Data Mining layer comprises a set of specific services useful to prepare and execute a data mining application, as well as present its results. The system has not been yet implemented on a Grid; a preliminary fully centralized version of the system is currently available.

GATES (Grid-based AdapTive Execution on Streams) is an OGSA based system that provides support for processing of data streams in a Grid environment [1]. GATES aims to support the distributed analysis of data streams arising from distributed sources (e.g., data from large scale experiments/simulations), providing automatic resource discovery, and an interface for enabling self-adaptation to meet real-time constraints.

Finally, we outline here some interesting data mining testbeds developed at the National Center for Data Mining (NCDM) at the University of Illinois at Chicago (UIC) [4]:

- *The Terra Wide Data Mining Testbed* (*TWDM*). TWDM is an infrastructure for the remote analysis, distributed mining, and real time exploration of scientific, engineering, business, and other complex data. It consists of five geographically distributed nodes linked by optical networks through *StarLight* (an advanced optical infrastructure) in Chicago. These sites include StarLight itself, the Laboratory for Advanced Computing at UIC, SARA in Amsterdam, and the Dalhousie University in Halifax. A central idea in TWDM is to keep generated predictive models up-to-date with respect to newly available data, in order to achieve better predictions (as this is an important aspect in many *critical* domains, such as infectious disease tracking). TWDM is based on *DataSpace*, another NCDM project for supporting real-time streaming data; in DataSpace the *Data Tranformation Markup Language* (*DTML*) is used to describe how to update *profiles*, i.e., aggregate data which are inputs of predictive models, on the basis of new *events*, i.e., new bits of information.

- *The Terabyte Challenge Testbed.* The Terabyte Challenge Testbed is an open, distributed testbed for DataSpace tools, services, and protocols. It involves a number of organizations, including the University of Illinois at Chicago, the University of Pennsylvania, the University of California at Davis and the Imperial College. The testbed consists of ten sites distributed over three continents connected by high-performance links. Each site provides a number of local clusters of workstations which are connected to form wide area *meta-clusters* maintained by the *National Scalable Cluster Project*. So far, meta-clusters have been used by applications in high energy physics, computational chemistry, nonlinear simulation, bioinformatics, medical imaging, network traffic analysis, digital libraries of video data, etc. Currently, the Terabyte Challenge Testbed consists of approximately 100 nodes and 2 terabytes of disk storage.

[4]http://www.ncdm.uic.edu/testbeds.htm.

- *The Global Discovery Network* (*GDN*). The GDN is a collaboration between the Laboratory for Advanced Computing of the National Center for Data Mining and the Discovery Net project. It will link the Discovery Net to the Terra Wide Data Mining Testbed to create a combined global testbed with a critical mass of data.

Some of the systems discussed above support specific domains applications, others support a more general class of problems. Moreover, some of such systems are mainly advanced interfaces for integrating, accessing, and elaborating large datasets, whereas others provide more specific functionalities for the support of typical knowledge discovery processes.

5 Conclusions

In this report we have addressed the main problems and discussed several proposals concerning knowledge extraction from distributed and stream data sources. We plan to investigate these issues in the field of the extraction of frequent patterns, like itemsets, sequences, and closed ones. We intend to define and evaluate a distributed framework for DDM, based on the merging of partial results extracted from distributed sites, by using interpolation to replace missing patterns and associated supports. We also plan to adapt and extend this framework to mine multiple and distributed stream sources.

Moreover, in this report we have also addressed the issues in exploiting some recent distributed infrastructures like the computational Grid, and surveyed some important proposals. In this regard, we plan to investigate how to define and compose Grid services for implementing distributed knowledge discovery and data mining services on SOA-Grids, i.e. the recent SOA-based Grid middlewares. In particular, we are interested in defining Grid services for searching Grid resources, composing software and data elements, and manage the execution of the resulting data mining application on a Grid.

References

[1] G. Agrawal. High-level interfaces and abstractions for grid-based data mining. In *Workshop on Data Mining and Exploration Middleware for Distributed and Grid Computing*, September 2003.

[2] R. Agrawal and J.C. Shafer. Parallel mining of association rules. In *IEEE Transaction On Knowledge and Data Engineering*, 1996.

[3] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16. ACM Press, 2002.

[4] F. Berman. From teragrid to knowledge grid. *Communitations of the ACM*, 44(11):27–28, 2001.

[5] P. Bonnet, J.E. Gehrke, and P. Seshadri. Towards sensor database systems. In *Proc. of the 2nd International Conference on Mobile Data Management.*, 2001.

[6] M. Cannataro, D. Talia, and P. Trunfio. Knowledge grid: High performance knowledge discovery services on the grid. In *Proc. of the 2nd Int. Workshop on Grid Computing (Grid 2001)*, 2001.

[7] Mario Cannataro and Domenico Talia. The knowledge grid. *Communitations of the ACM*, 46(1):89–93, 2003.

[8] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *ICALP '02: Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, pages 693–703, London, UK,, 2002. Springer-Verlag.

[9] R. Chen, K. Sivakumar, and H. Kargupta. Collective Mining of Bayesian Networks from Distributed Heterogeneous Data. *Knowl. Inf. Syst.*, 6(2):164–187, 2004.

[10] Cheung, Han, Ng, Fu, and Fu. A fast distributed algorithm for mining association rules. In *PDIS: International Conference on Parallel and Distributed Information Systems*. IEEE Computer Society Technical Committee on Data Engineering, and ACM SIGMOD, 1996.

[11] G. Cormode and S. Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. In *PODS '03: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 296–306. ACM Press, 2003.

[12] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[13] V. Curcin, M. Ghanem, Y. Guo, M. Kohler, A. Rowe, J. Syed, and P. Wendel. Discovery net: Towards a grid of knowledge discovery. In *Proc. of the 8th Int. Conference on Knowledge Discovery and Data Mining (KDD 2002)*, 2002.

[14] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and R. Kargupta. Distributed data mining in peer-to-peer networks. *IEEE Internet Computing*, 2006.

[15] E.D. Demaine, A. López-Ortiz, and J.I. Munro. Frequency estimation of internet packet streams with limited space. In *ESA '02: Proceedings of the 10th Annual European Symposium on Algorithms*, pages 348–360, London, UK,, 2002. Springer-Verlag.

[16] Thomas G. Dietterich. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning*, 40(2):66–75, 2000.

[17] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proc. 2000 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'00)*, pages 71–80, 2000.

[18] George Forman and Bin Zhang. Distributed data clustering can be efficient and exact. *SIGKDD Explor. Newsl.*, 2(2):34–38, 2000.

[19] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. *The Physiology of the Grid*, volume Grid Computing: Making the Global Infrastructure a Reality, pages 217–249. Wiley, 2003.

[20] N. Giannadakis, A. Rowe, M. Ghanem, , and Y. Guo. Infogrid: Providing information integration for knowledge discovery. *Information Sciences*, 155(3):199–226, 2003.

[21] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515–528, 2003.

[22] E-H.S. Han, G. Karypis, and V. Kumar. Scalable parallel data mining for association rules. In *IEEE Transaction on Knowledge and Data Engineering*, 2000.

[23] T. Hinke and J. Novonty. Data mining on nasa's information power grid. In *Proc. of the 9th Int. Symposium on High Performance Distributed Computing*, 2002.

[24] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106. ACM Press, 2001.

[25] R. Jin and G. G. Agrawal. An algorithm for in-core frequent itemset mining on streaming data. IEEE ICDM'05, 2005.

[26] W. E. Johnston. Computational and data grids in large-scale science and engineering. *Future Generation Computer Systems*, 18(8):1085–1100, 2002.

[27] H. Kargupta and P. Chan. *Advances in Distributed and Parallel Knowledge Discovery*. AAAI Press, 2000.

[28] R.M. Karp, S. Shenker, and C.H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems (TODS)*, 28(1):51–55, 2003.

[29] Y.-N. Law and C. Zaniolo. An adaptive nearest neighbor classification algorithm for data streams. In *Proc. of PKDD 2005, LNAI 3721*, pages 108–120. Springer, 2005.

[30] G. Manku and R. Motwani. Approximate frequency counts over data streams. In *In Proceedings of the 28th International Conference on Very Large Data Bases*, August 2002.

[31] Misra.J. and D. Gries. Finding repeated elements. Technical report, Ithaca, NY, USA,, 1982.

[32] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-data algorithms for high-quality clustering. In *Proc. of . In ICDE'02*, 2002.

[33] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[34] B. Park and H. Kargupta. Distributed Data Mining: Algorithms, Systems, and Applications. In *Data Mining Handbook*, pages 341–358. IEA, 2002.

[35] A. Prodromidis and P. Chan. Meta-learning in distributed data mining systems: Issues and Approaches. In *Advances of Distributed Data Mining*. AAAI Press, 2000.

[36] A. Schuster and R. Wolff. Communication Efficient Distributed Mining of Association Rules. In *ACM SIGMOD*, Santa Barbara, CA, April 2001.

[37] D. Skillicorn and D. Talia. Mining large data sets on grids: Issues and prospects. *Computing and Informatics*, 21(4):347–362, 2002.

[38] R. Wolff and A. Schuster. Mining Association Rules in Peer-to-Peer Systems. In *The Third IEEE International Conference on Data Mining (ICDM'03)*, Melbourne, FL, November 2003.

[39] Xiaowei Xu, Jochen J&#228;ger, and Hans-Peter Kriegel. A fast parallel clustering algorithm for large spatial databases. *Data Min. Knowl. Discov.*, 3(3):263–290, 1999.

[40] M.J. Zaki. Parallel and distributed association mining: A survey. In *IEEE Concurrency*, 1999.