# *Distributed mining of molecular fragments*

Proc. of IEEE DMGrid 2004, Workshop on Data Mining and Grid of IEEE ICDM 2004, Int. Conf. on Data Mining, Brighton (UK), 1-4 Nov., 2004

# Distributed Mining of Molecular Fragments

Giuseppe Di Fatta
ICAR-CNR, Consiglio Nazionale delle Ricerche
Viale delle Scienze, 90128 Palermo, Italy
and Konstanz University, Germany
difatta@pa.icar.cnr.it

Michael R. Berthold
Konstanz University,
Dept. of Computer and Information Science
78457 Konstanz, Germany
Michael.Berthold@uni-konstanz.de

## Abstract

*In real world applications sequential algorithms of data mining and data exploration are often unsuitable for datasets with enormous size, high-dimensionality and complex data structure. Grid computing promises unprecedented opportunities for unlimited computing and storage resources. In this context there is the necessity to develop high performance distributed data mining algorithms. However, the computational complexity of the problem and the large amount of data to be explored often make the design of large scale applications particularly challenging. In this paper we present the first distributed formulation of a frequent subgraph mining algorithm for discriminative fragments of molecular compounds. Two distributed approaches have been developed and compared on the well-known National Cancer Institute's HIV-screening dataset. We present experimental results on a small-scale computing environment.*

## 1. Introduction

The main goal of life sciences research is the discovery of new drugs. All major pharma companies rely on the introduction of several new, highly successful new drugs (so-called blockbuster medications) per year in order to finance the extremely expensive and lengthy drug discovery process. Typically, it takes more than 10 years until a newly identified drug candidate reaches the market. Despite advances in the analysis of the human genome and better understandings of the underlying biological interactions, one crucial step in drug discovery remains the so-called High Throughput Screening and the subsequent analysis of the generated data. In this screening, hundreds of thousands of potential drug candidates are automatically tested for a desired activity, such as blocking a specific binding site or attachment to a particular protein. This activity is believed to be connected to, for example, the inhibition of a specific disease. Once all these candidates have been automatically screened it is necessary to concentrate on a few hundred promising candidates for further, more careful (and cost-intensive) analysis. Many tools concentrate on techniques that allow the biochemists to explore the results of the screening analysis, to determine which molecules to investigate further. This step is crucial for the success of the entire drug discovery process. Losing a potential blockbuster drug here can result in a loss of up to one billion euro at a later stage. A promising approach focuses on the analysis of the molecular structure and the extraction of pieces of these molecules that are correlated with activity. These so-called *discriminative fragments* can then be used to directly identify groups of promising molecules by the user because of the representation, which is immediately understandable to chemists and biologists. Discriminative molecular fragment discovery can be formulated as a frequent subgraph mining (FSM) problem [16] in analogy to the association rule mining (ARM) problem [2, 19]. While in ARM the main structure of the data is a list of items (itemset) and the basic operation is the subset test, in FSM graph and subgraphs isomorphism is the intrinsic nature of the problem.

Sequential algorithms are limited by single processor computing resources and are often unsuitable for extremely large datasets (millions of molecules) and unlimited size of the fragments that can be discovered. Quite obviously, parallel approaches to this type of problem are a promising alternative to the current sequential algorithms, both with respect to storage and time limitations.

Recently, undergoing efforts for Grid computing middleware development [10, 11] have promised unprecedented opportunities for unlimited computing and storage resources. In this context it is necessary to develop high performance distributed data mining algorithms.

A Grid environment [9, 13] provides high performance computing facilities and transparent access to them in spite of their remote location, different administrative domains and hardware and software heterogeneous characteristics. A Grid is a combination of distributed and heterogeneous computing, storage and communication resources for executing large-scale applications. A distinction is sometimes made between Data Grids and Computational Grids. Computational Grids normally deal with large-scale computationally intensive problems on small data sets, while Data Grids deal with large-scale data-intensive problems on large amounts of data, i.e. typical data mining problems. In the context of molecular fragment analysis both aspects are present. This makes the effective exploitation of a large-scale computational and storage system a very complex task.

In this paper we present the first distributed formulation for the frequent subgraph mining problem. The algorithm has been applied to the analysis of a set of real molecular compounds, the well-known National Cancer Institute's HIV-screening dataset.

The rest of this paper is structured as follows. In the next section we discuss related problems and approaches to the molecular fragment mining problem. In section 3 we define the problem and briefly describe a serial algorithm on which our distributed approach is based. In section 4 we present two distributed implementations of the sequential mining algorithm. Section 5 describes the experiments we conducted to verify the performance of the distributed approaches. Finally, we provide concluding remarks.

## 2. Related works

A number of approaches to find discriminative molecular fragments have recently been published [8, 18, 4] but they are all limited by the complexity of the underlying problem. Finding frequent subgraphs in a set of graphs is computationally extremely expensive. The subgraph isomorphism test is known to be an NP-complete problem for general graphs. Some of these algorithms can therefore operate on very large molecular databases but only find small fragments [8, 18] whereas others can find larger fragments but are limited by the maximum number of molecules they can analyse [4, 15].

Finding discriminative fragments in a set of molecules can be seen as analysing the space of all possible fragments, that is all subgraphs that can be found in the entire molecular database. Obviously this set of all existing fragments is enormous, a single molecule of average size can already contain in the order of hundred thousand different fragments. Existing methods to find discriminative fragments usually organize the space of all possible fragments in a lattice, which models subgraph relationships, that is, edges connect fragments that differ by exactly one atom and/or bond. The search then reduces to traversing this lattice and reporting all fragments that fulfil the desired criteria. Based on existing data mining algorithms for market basket analysis [2, 19] these methods conduct depth-first [4] or breadth-first searches [18, 8].

None of these algorithms in a single processor can be used for extremely large datasets (millions of molecules) and unlimited size of the fragments that can be discovered. Quite obviously, parallel approaches to this type of problem are a promising alternative to the current sequential algorithms. In recent years several parallel and distributed algorithms have been proposed for the association rule mining problem (D-ARM) [20]. However, currently no parallel and distributed FSM algorithms have been proposed in the literature. Distributing such search algorithms on parallel resources is non-trivial. The complexity of the problem and the large amount of data to be explored make parallel formulations of these methods extremely challenging as we will discuss later in more detail.

## 3. Molecular fragment mining (MoFa)

The problem of selecting discriminative molecular fragments in a set of molecules can be formulated in terms of frequent subgraph mining in a set of graphs. Molecules are represented by attributed graphs, in which each vertex represents an atom and each edge a bond between atoms. Each vertex carries attributes that indicate the atom type (i.e., the chemical element), a possible charge, and whether it is part of a ring. Each edge carries an attribute that indicates the bond type (single, double, triple, or aromatic). Frequent molecular fragments are subgraphs that have a certain minimum support in a given set of graphs, i.e., are part of at least a certain percentage of the molecules. However, in order to restrict the search space, only connected substructures, i.e., graphs having only one connected component, are considered. Discriminative molecular fragments are contrast substructures, which are frequent in a predefined set of molecules and infrequent in the complement of this subset. In this case two parameters are required: a minimum support ($minSupp$) for the focus subset and a maximum support ($maxSupp$) for the complement.

The distributed approach presented in this paper is based on the sequential algorithm (MoFa) described in [4]. The algorithm organizes the space of all possible fragments in an efficient search tree. An example of such a search tree is depicted in Figure 1. Each possible subgraph of the molecular structures is evaluated in terms of the number of embeddings that are present in the molecular database.

The algorithm is based on an exhaustive depth-first search strategy. Each node of the search tree represents a candidate frequent fragment. A search tree node evaluation comprises the generation of all the embeddings of the fragment in the molecules. The embedding list allows both a fast computation of the fragment support in the ac-

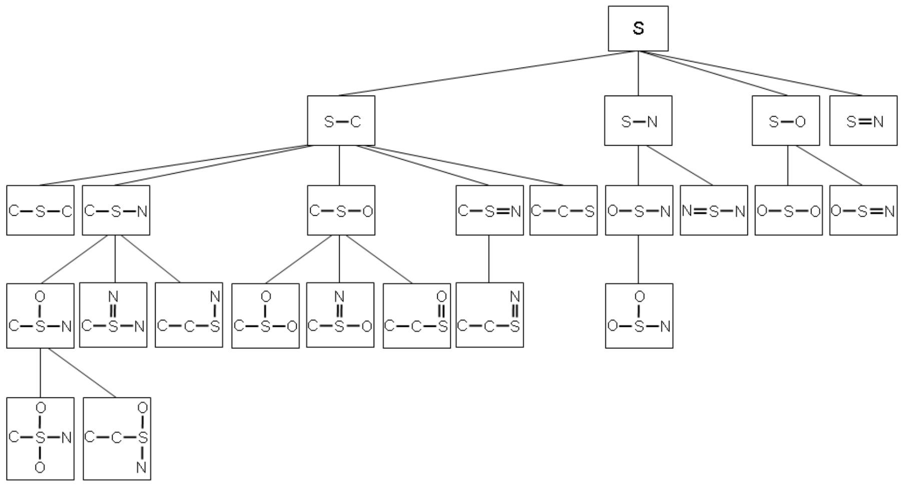
**Figure 1. Molecular fragment search tree**

tive and inactive molecules and a fast extension to bigger fragments. When a fragment meets the minimum support criterion, it is extended by one bond to generate new search tree nodes. When the fragment meets both criteria of minimum support in active molecules and maximum support in the inactive molecules, it is then reported as a discriminative frequent fragment. Moreover, those frequent fragments which have the same support values are further filtered. Fragments, which are subgraphs of other frequent fragments having equal support values, are not considered interesting and discarded.

The search starts from a root node with a single atom and it is iterated for each frequent atom type. The algorithm prunes the DFS tree according to three criteria. The support-based pruning exploits the anti-monotone property of fragment support. The size-based pruning exploits the anti-monotone property of fragment size. And, finally, a partial structural pruning is based on a local order of atoms and bonds. For further details on the algorithm we refer to [4].

## 4. Distributed molecular fragment mining

Parallel and distributed approaches are a promising alternative to the current sequential algorithms. Sequential algorithms cannot provide scalability in terms of data size and dimensionality, neither better quality of the results (wider range for user parameters) nor dramatically better running time performance. In this section we present the first distributed approach for the molecular fragment mining that is based on the sequential algorithm described in the previous section.

In the first distributed approach we are presenting, we adopt a search space partitioning strategy along with a dynamic load balancing (DLB). Thereafter, we introduce a second approach whereby we combine the search space partitioning with a data partitioning strategy.

Our ultimate target architecture is a large-scale multi-domain computational environment, i.e. a Grid infrastructure. So, our general approach is to partition the problem into independent subtasks in order to minimize communication and synchronization among processors. Although we do not address the issue in this work, independent tasks would also allow the introduction of fault tolerance mechanisms more easily than in distributed applications with a complex communication pattern.

For this reason we have adopted partitioning strategies and discarded solutions based on a collaborative approach among processes, such as distribution techniques similar to ones proposed in [3, 12], which are more suited for single-domain environments like parallel machines, networks of workstations and hybrid/hierarchical systems. Moreover, all the algorithms that have been proposed for D-ARM in the past years assume a homogeneous and dedicated environment and do not provide dynamic load balancing [20].

### 4.1. Search space partitioning

Partitioning a DFS tree has been widely and successfully adopted in many applications. In general, it is quite straightforward to prune the search tree to generate a new independent job (Figure 2), which can be assigned to an idle processor. In this case no synchronization is required among remote jobs.

Following this approach each worker has local access to the entire dataset. This allows each miner to execute the same algorithm of the serial approach. A job assignment contains the description of the search node pruned at the donor worker. It must contain all the information needed to restart the search from exactly the same point in the search space, which includes the node state to rebuild the same local order necessary to prune the search tree as in the sequential algorithm (cfr. structural pruning in [4]). Furthermore, the donor worker knows the exact subset of molecules in which the fragment represented by the pruned node can actually be embedded. When the cardinality of such a subset is not too large, a list of the molecule IDs will be added to the job description. In this case the receiver worker can load only the relevant molecules from the local database (DB selection optimisation).

During the exploration of the search tree the embedding list in both active and inactive molecules allows the algorithm to avoid computationally expensive re-embeddings for frequency counting in order to perform the $minSupp$ and $maxSupp$ tests. The list of the embeddings in the molecules allows also a fast selection (filtering) of the interesting frequent fragments. Substructures of frequent fragments are not interesting and not reported. The latter filtering step is incomplete in the case of a search tree partitioning. Each worker maintains only a local and partial list of substructures found during the execution of subtasks. Therefore, when the master node merges the partial lists of frequent fragments reported at the completion of each subtask, it has to filter out all the equivalent fragments and all the fragments that are substructures of other fragments as well. This results in extra computation required by the distributed algorithm. In order to avoid this computation overhead we would have introduced extra communication and synchronization. However, this computation overhead is not very relevant because the cardinality of the list of candidate fragments is typically of several orders of magnitude smaller than the cardinality of the overall dataset.

A static partition of the search space can only be adopted when job running times can be estimated. In our application we cannot estimate the complexity of subtasks. For such irregular problems it is essential to provide a dynamic load balancing. We adopted a search tree partitioning with a self-adaptive job-granularity based on dynamic load balancing, which is discussed in the next section.

### 4.2. Dynamic load balancing

The dynamic load balancing (DLB) requirement of our problem is related to the same requirement of several other irregular problems based on a tree structure, e.g. problems solved using the divide and conquer strategy and other problems based on a search tree. Several DLB techniques for irregular problems are based on assumptions on task times [14, 6], or the availability of workload estimations [7]. In [14] uniform time tasks are assumed. In [6] it is assumed that the smallest task time is comparable to or greater than network communication time for a task. In [7] the computation, first, is evenly partitioned among processors and, successively, task migration is adopted to maintain load balance in the system.

Unfortunately, in our application none of these assumptions can be made. We cannot even provide minimum or maximum bounds for the running time of subtasks. It is quite challenging to efficiently parallelize irregular problems with such an unpredictable workload.

Moreover, in a large-scale multi-domain environment we also have to deal with the heterogeneous and dynamic load of processors and networks.

The DLB approach we adopted for the tests in a small-scale computational environment is based on a centralized job pool and the Master-Workers paradigm. The master operates as job manager and starts a local worker with the root node of the entire search tree. Whenever there is an idle processor the job manager solicits a worker to prune part of its search tree and spawn a new job description. The job manager collects new jobs and allocates them to idle processors upon request.

The job manager requests the generation of new jobs to workers in order to maintain a minimum level in the job pool. Two thresholds, $minJobs$ and $maxJobs$ determine the range in which the job manager queries a worker to spawn a new job. (In our tests these values have been set to the number of processors and twice this number, respectively.) Whenever a worker terminates a job, it sends a request for a new one. Each worker keeps also a local pool of unprocessed jobs. At the completion of a job, the request and reception of a new job can be overlapped to the execution of a job from the local pool. Overlapping computation with job assignment avoids most of the communication overheads. Only the latency of the first job assignment and the last report cannot be avoided. (In our tests we adopted a single job buffer in each local worker.)

In problems with uniform or bounded task times the generation of either too small or too big jobs is not an issue. In our case wrong job granularity may decrease the efficiency and limit the maximum speedup. While a coarse job granularity may induce load imbalance and bounds on the maximum speedup, a fine granularity may decrease the distributed system efficiency and more processing nodes will be required to reach the maximum speedup. Thus, it is important to provide an adaptive mechanism to find a good trade off between load balancing and job granularity.

In order to accomplish this aim we introduce two simple mechanisms to avoid the generation of trivial tasks and processor idling. The worker follows three rules to prune a search node. The local worker must have done some work before pruning a node and must still have some work to do after pruning. Moreover, the new job should have enough

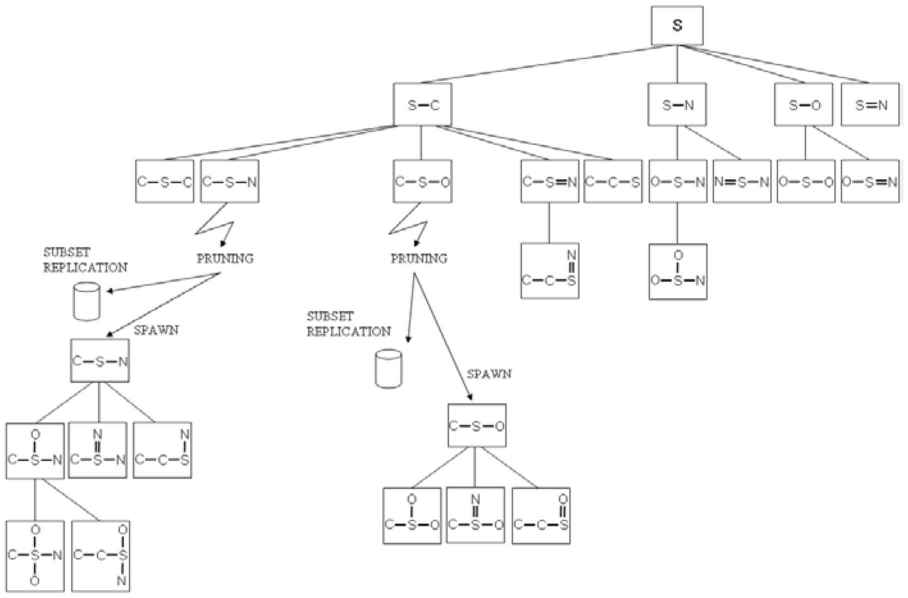

**Figure 2. Search tree partitioning**

support in the active compounds, thus avoiding spawning a trivial job. A search tree node $n$ can be pruned only if

1. $depth(n) >= minDepth$,
2. $support(n) >= ((1 + alpha) * minSupp)$, and
3. $sibling(n) >= minSib$

where $alpha$ is a tolerance factor and $sibling(n)$ specifies the number of sibling nodes of $n$. The values of these parameters are not critical and in our experiments we adopted $minDepth = 1$ (with $depth(root\_node) = 0$), $alpha = 0.1$ and $minSib = 1$. These rules for pruning the search tree guarantee that the local worker does not run out of work while donating non-trivial part of its task.

The job manager has in charge the selection of the task to be further partitioned to generate new jobs to be assigned to idle workers. Partitioning small subtrees clearly decreases the overall performance of the system. Then, it would be useful to know which task is the most complex among the assigned tasks. But, as already mentioned, we cannot estimate the complexity of a task and its running time. However, we can keep track of the order in which jobs have been assigned and the job manager, when necessary, simply solicits the partitioning of the longest lived job. This choice can be motivated by two reasons. The oldest assigned job is likely to be among the most complex ones. And this probability increases over time. Second, a long job execution time may also depend on the heterogeneity of the processing nodes and their loads. With such a choice we provide help to the node which is likely to be overloaded either by the current mining task assignment or by other unrelated processes.

This approach guarantees a good trade-off between job granularity and load balancing independently from the subtask size distribution and our ability to estimate it, and it is suitable for heterogeneous computing infrastructures.

However, when the number of nodes increases, the centralized job manager will obviously become the bottleneck of the system. In this case the DLB framework has to be extended with a distributed job management approach. In our current implementation each worker receives and donates jobs, but does not play any role in the job allocation process. Either a hierarchical or a completely distributed management of the job pool would provide the scalability needed to run the application in a large scale environment. Our preliminary experiments have been carried out on a small cluster of workstations and we leave this extension for future research.

## 4.3. Data partitioning

When data-locality cannot be guaranteed, we have to partition the data among the processors. For instance, a very large size of the dataset and a high number of processing nodes do not make the entire database replication feasible. We address this problem by taking into account the specific application domain. The number of active compounds is typically much smaller than the number of the inactive ones. Thus, we can still duplicate the focus dataset (active molecules subset) in each node and partition only the complement dataset (inactive molecules subset). The second distributed algorithm we present, combines the space search partitioning with the complement dataset partitioning.

In a first phase, a search space partition algorithm is executed on the focus dataset only. The molecules of the local partition of the complement dataset are not used during this first step. A list of candidate fragments is incrementally built at the centralized manager node from the partial list reported by the workers. When lists of candidates are merged, duplicates are discarded. Then, in a second phase, the complete list of candidate fragments is broadcasted to all workers. Each worker computes partial counts of the frequency of the candidates in the local partition of the complement dataset. It should be noticed that in this step several subgraph isomorphism tests are required to determine if a candidate fragment is a substructure of the inactive molecules. The optimisation based on the embedding list to avoid this test adopted by serial MoFa and by the previous distributed approach cannot be applied. Finally, the partial counts are collected and the list of candidates is filtered by testing the $maxSupp$ on the complement dataset.
Without access to the entire set of inactive molecules, the task of filtering the discriminative fragments has to be performed in a second phase after the entire search tree has been explored using only the focus dataset.

The number of subgraph isomorphism tests which are performed potentially makes the second phase very slow. However, the number of tests is proportional to the number of candidates (typically orders of magnitude less the dataset cardinality) and the partition size, which depends on the number of processors. As a consequence, this second approach based on both search space and data partitioning is expected to exhibit good scalability characteristics and may be suited for a large-scale computing environment. Furthermore, the different resource requirements between the first phase and the second phase make this approach benefit from a Grid computing infrastructure with dynamic management of computing resources, while it is less suitable for parallel architecture and traditional distributed systems (e.g., NoW).

## 5. Experimental results

The two distributed algorithms (denoted by P1 and P2) have been tested for the analysis of a set of real molecular compounds, a well-known, publicly available dataset from the National Cancer Institute, the DTP AIDS Antiviral Screen dataset [1]. This screen utilized a soluble formazan assay to measure protection of human CEM cells from HIV-1 infection [17]. Compounds able to provide at least 50% protection to the CEM cells were retested. Compounds that provided at least 50% protection on retest were listed as moderately active (CM). Compounds that reproducibly provided 100% protection were listed as confirmed active (CA). Compounds not meeting these criteria were listed as confirmed inactive (CI). We used 37169 total compounds, of which 325 belong to class CA, 875 are of class CM and the remaining 35969 are of class CI. In order to carry out tests on different sizes of focus and complement datasets we combined these compounds as follows. We joined the CA set with a different number of CM compounds (0, 325, 650, 975) to form four focus datasets of size $f1 = 325$, $f2 = 650$, $f3 = 975$, $f4 = 1200$. We used the CI compounds to form four complement datasets of size $c1 = 9000$, $c2 = 18000$, $c3 = 27000$, and $c4 = 35969$.

Experimental tests have been carried out in a local cluster of ten computing nodes[1]; the software has been developed in Java. The communication among processes has been implemented using an MPI-Java wrapper (mpi-Java v1.2.5 [5]) to access a native MPI library (MPICH v1.2.5.2). The Globus toolkit v2.4.3 has been adopted to provide a Grid middleware infrastructure. In spite of the small-scale single-domain computing environment we explicitly wanted to adopt a Grid infrastructure to demonstrate the feasibility of a Grid approach to the problem and its effectiveness.
In the next sections we present a performance analysis of the serial algorithm [4] and the performance evaluation of the two distributed approaches proposed in this work.

### 5.1. Serial algorithm performance

This section presents an experimental analysis of the sequential algorithm to demonstrate the potential benefits of the distributed approach. Two sets of tests have been carried out and the results are shown, respectively, in figures 3 and 4. In these tests we fixed the maximum support in the inactive compounds at $1\%$ and changed the minimum support in the active compounds in the range $5 - 20\%$.

Figure 3 shows how running time varies for different values of the minimum support in the active compounds. When a lower value is chosen, the algorithm has to explore further and deeper branches of the search tree and bigger fragments. This test shows the exponential growth of the running time when both the minimum support and the number of active molecules (the different curves) decrease. The number of active compounds is relatively small compared to the total number of molecules in the dataset. Their number does not have a significant influence on the overall size of the dataset. Rather, for a fixed percentage value of the minimum support, the minimum absolute

[1]Nine nodes are equipped with a cpu Intel Xeon 2.40GHz and 3GB RAM. One node is equipped with a cpu Intel Pentium M 1.7GHz and 1GB RAM. All run Linux 2.4.21 and Java SE 1.4.2.

number of active compounds, which defines the frequent fragments, decreases. As a consequence, further and deeper branches of the search tree have to be explored. The Serial algorithm could not complete the test with f1=325 and $minSupp = 5\%$ due to an *OutOfMemory* error. DFS's memory requirements are proportional to the maximum depth of the search tree. In this case the combination of a small value of the parameter $minSupp$, a small number of compounds in the focus dataset, and a high number of compounds in the complement dataset caused the algorithm to reach the memory limit of a single processing node. Hence, the point (5, 2010) in the chart of figure 3 is only a lower bound for the running time of the serial algorithm.

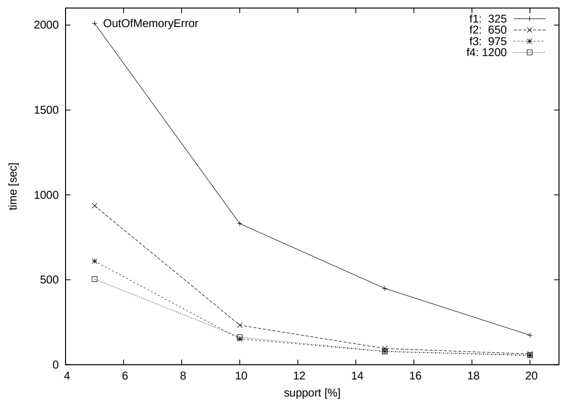

**Figure 3. Support influence on running times for different number of active (CA and CM) compounds ($c = 35969$)**

Figure 4 shows the relation between the number of inactive compounds, hence approximately the dataset size, the minimum support and the running time. In this case the equal distance among the different curves indicates a linear relation between running time and dataset size for a given minimum support.

Several techniques can be adopted to improve the performance of the algorithm. One, which is worth mentioning, is the adoption of knowledge from the specific application domain. Significant reduction of the search space, for instance, can be obtained by considering aromatic rings of carbon atoms as a single macro-node in the graph representation of the molecules. For simplicity, in this work we do not consider this or other optimisation techniques.

Furthermore, the analysis of the sequential algorithm also pointed out the irregular nature of the search tree. An irregular problem is characterized by a highly dynamic or unpredictable domain. In this application the complexity and the exploration time of the search tree, and even of a single search tree node cannot be estimated. The data mining nature of the problem makes the time required to visit a node unpredictable. In our tests a single node exploration can take from few milliseconds to several seconds. Moreover, depth and fan of the search tree is also unpredictable.

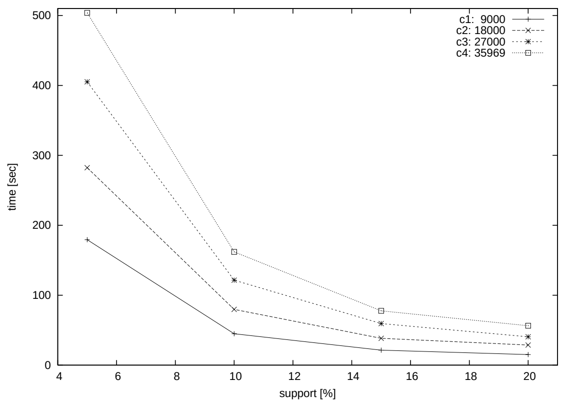

**Figure 4. Support influence on running times for different number of inactive (CI) compounds ($f = 1200$)**

## 5.2. Distributed algorithms evaluation

Of course, the first benefit we expect from distributing a computational load in several nodes is a decrease in the running time. This is even more important in a mining application like the discriminative fragment discovery. Very often the user does not know in advance what to expect from the mining activity on a dataset. Moreover, the choice of proper values for the application parameters may require several trials. Short response times in an interactive session are extremely important. Secondly, more and more often the amount of available data overcomes the computing throughput of a single processing node. In this case the search can only be accomplished on a small subset of the data, thus reducing the quality of the results. Furthermore, in order to adhere to the computational constraints the user has to limit the exploration of the range of parameters (e.g. $minSupp$).
Figures 5 and 6 show the running times of the serial algorithm, of the two distributed algorithms, namely P1 and P2, and of the first phase of P2 (P2-1) when the focus and complement dataset sizes are varied. In the tests of Figures 5 we adopted all available inactive compounds ($c = 35696$). In the tests of Figures 6 we adopted 1200 active compounds. In both cases we chose $minSupp = 5\%$ and $maxSupp = 1\%$, and all available computing nodes are used to run the distributed algorithms. P2-1 running time measures correspond to the time required to explore the entire search tree and to produce a list of candidates

to be filtered in the second phase. From these two charts it becomes evident that P1 outperforms P2. The second phase, which processes the local partition of the complement dataset, dominates the running time of P2, while the first phase (P2-1) runtime behaviour is equivalent to P1's runtime - just scaled down by the reduced number of compounds.

Finally, in figure 7 we show the speedup of the algorithms. In absolute value the algorithm P1 exhibits better speedup than P2. Nevertheless, in spite of the limited number of processors used for our test, P1's speedup already shows a sub-linear growth. P2's speedup improves with a higher number of processors and may provide better scalability properties than P1 for large-scale systems. Although these results are obviously preliminary they are highly promising. Further investigation is required to better evaluate the performance of the distributed approaches in a large-scale computational environment.

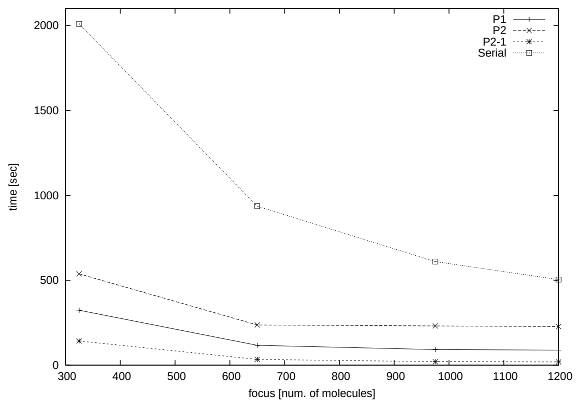

**Figure 5. Running time comparison for a different number of active compounds**

## 6. Conclusions

Grid computing infrastructures are foreseen to provide an unprecedented opportunity for very demanding applications in terms of computing and storage requirements. High performance distributed computing is becoming an essential component in data mining and data exploration. In this paper we presented the first distributed formulation for the frequent subgraph mining problem applied to the task of discriminative molecular fragment discovery. Two algorithms have been designed following a partitioning criterion of the search and data space. Very low communication and synchronization for both algorithms and potentially very high scalability for the second make these data mining algorithms good candidates for Grid computing consumers.

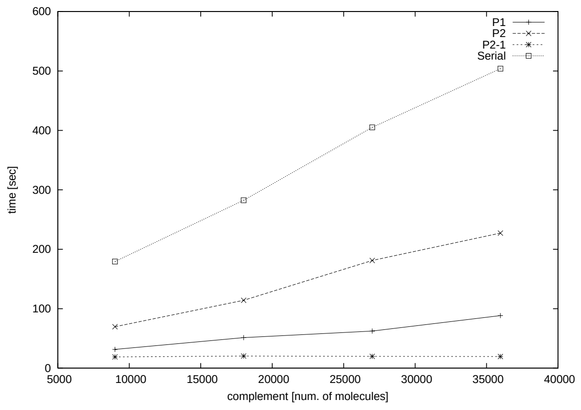

**Figure 6. Running time comparison for a different number of inactive compounds**

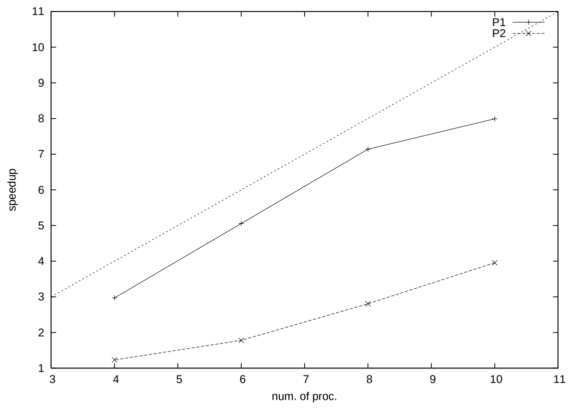

**Figure 7. Speedup (f=650 with minSupp=5%, c=35696 with maxSupp=1%)**

Several issues have been discussed for an effective design of a large-scale distributed FSM algorithm. Tests on a set of real compounds in a small-scale computing environment provided encouraging preliminary results.

## 7. Acknowledgements

The research has been made possible by support from the Deutscher Akademischer Austausch Dienst (DAAD) and by a joint support from the Deutsche Forschungsgemeinschaft (DFG) and Italian National Research Council (CNR). We also thank the Department of Information and Computer Science of the University of Konstanz for the use of their machines.

## References

[1] http://dtp.nci.nih.gov/docs/aids/aids_data.html.

[2] R. Agrawal, T. Imielienski, and A. Swami. Mining association rules between sets of items in large databases Proc. of Conf. on Management of Data. pages 207–216.

[3] R. Agrawal and J. Shafer. Parallel mining of association rules. *IEEE Trans. Knowledge and Data Eng.*, 8(6):962969, Dec. 1996.

[4] C. Borgelt and M. R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. IEEE International Conference on Data Mining (ICDM 2002, Maebashi, Japan). pages 51–58, December 09-12, 2002.

[5] B. Carpenter, G. Fox, S.-H. Ko, and S. Lim. mpijava 1.2: Api specification.

[6] S. Chakrabarti, A. Ranade, and K. Yelick. Randomized load-balancing for tree-structured computation, In Scalable High Performance Computing Conference, Knoxville, TN. 1994.

[7] Y. Chung, J.-W. Park, and S.-H. Yoon. An asynchronous algorithm for balancing unpredictable workload on distributed-memory machines. *ETRI Journal*, 20(4):346–360, Dec. 1998.

[8] M. Desphande, M. Kuramochi, and G. Karypis. Automated approaches for classifying structures Proc. of Workshop on Data Mining in Bioinformatics (BioKDD). pages 11–18, 2002.

[9] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International J. Supercomputer Applications*.

[10] Globus Project Team. The Globus Project, http://www.globus.org.

[11] A. Grimshaw and W. Wulf. The legion vision of a worldwide virtual computer. *In Communications of the ACM*, 40(1), January 1997.

[12] E.-H. Han, G. Karypis, and V. Kumar. Scalable parallel data mining for association rules. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):337–352, May/June 2000.

[13] I. Foster and C. Kesselman and J. Nick and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration Open Grid Service Infrastructure WG, Global Grid Forum. June 22, 2002.

[14] R. Karp and Y. Zhang. A randomized parallel branch-and-bound procedure, In Proceedings of the 20 Annual ACM Symp. on Theory of Computing. 1988.

[15] S. Kramer, L. de Raedt, and C. Helma. Molecular feature mining in hiv data Proc. of 7th Int. Conf. on Knowledge Discovery and Data Mining, (KDD-2001, San Francisco, CA). pages 136–143, 2001.

[16] T. Washio and H. Motoda. State of the art of graph-based data mining. *ACM SIGKDD Explorations Newsletter*, 5(1):59–68, July 2003.

[17] O. Weislow, R. Kiser, D. Fine, J. Bader, R. Shoemaker, and M. Boyd. New soluble formazan assay for hiv-1 cytopathic effects: Application to high flux screening of synthetic and natural products for aids antiviral activity. *Journal of the National Cancer Institute, University Press, Oxford, United Kingdom*.

[18] X. Yan and J. Han. gspan: Graph-based substructure pattern mining Proceedings of the IEEE International Conference on Data Mining ICDM, Maebashi City, Japan. 2002.

[19] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules Proc. of 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD'97). pages 283–296, 1997.

[20] M. J. Zaki. Parallel and distributed association mining: A survey. *IEEE Concurrency*, 7(4):14–25, 1999.