High Performance Subgraph Mining in Molecular Compounds

Giuseppe Di Fatta[1,2] and Michael R. Berthold[1]

[1] University of Konstanz, Dept. of Computer and Information Science, 78457 Konstanz, Germany
berthold@uni-konstanz.de
[2] ICAR, Institute for High Performance Computing and Networking, CNR, Italian National Research Council, 90018 Palermo, Italy
difatta@pa.icar.cnr.it

Abstract. Structured data represented in the form of graphs arises in several fields of the science and the growing amount of available data makes distributed graph mining techniques particularly relevant. In this paper, we present a distributed approach to the frequent subgraph mining problem to discover interesting patterns in molecular compounds. The problem is characterized by a highly irregular search tree, whereby no reliable workload prediction is available. We describe the three main aspects of the proposed distributed algorithm, namely a dynamic partitioning of the search space, a distribution process based on a peer-to-peer communication framework, and a novel receiver-initiated, load balancing algorithm. The effectiveness of the distributed method has been evaluated on the well-known National Cancer Institute's HIV-screening dataset, where the approach attains close-to linear speedup in a network of workstations.

1 Introduction

A crucial step in the drug discovery process is the so-called High Throughput Screening and the subsequent analysis of the generated data. During this process, 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 select few promising candidates for further, more careful and cost-intensive analysis. A promising approach focuses on the analysis of the molecular structure and the extraction of relevant molecular fragments that may be correlated with activity. Relevant molecular fragment discovery can be formulated as a frequent subgraph mining (FSM) problem [1] in analogy to the association rule mining (ARM) problem [2,3]. While in ARM the main structure of the data is a list of items (itemset) and the basic operation is the subset test, FSM is based on graph and subgraph isomorphism.

In this paper we present a high performance application of the frequent subgraph mining problem applied to the analysis of molecular compounds.

L.T. Yang et al. (Eds.): HPCC 2005, LNCS 3726, pp. 866–877, 2005.

The rest of the paper is structured as follows. In the next section we introduce the molecular fragment mining problem and discuss related approaches. In Sect. 3 we discuss alternative definitions of discriminative molecular fragments and briefly describe the sequential algorithm on which the distributed approach is based. In Sect. 4 and 5 we present, respectively, a high performance distributed computing approach for subgraph mining and the adopted dynamic load balancing policy. Section 6 describes the experiments we conducted to verify the performance of the distributed approach. Finally, we provide conclusive remarks.

2 Problem Definition and Related Works

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. 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.

These topological fragments carry important information and may be representative of those components in the compounds that are responsible for a positive behavior. Such discriminate fragments can be used to predict activity in other compounds [4] and to guide the synthesis of new ones.

A number of approaches to find frequent molecular fragments have recently been published [5,6,7,8] but they are all limited by the complexity of graph and subgraph isomorphism tests and by the combinatorial nature of the problem. Some of these algorithms can therefore operate on very large molecular databases but only find small fragments [5,6], whereas others can find larger fragments but are limited by the maximum number of molecules they can analyse [7,8].

Finding frequent fragments in a set of molecules can be seen as analysing the space of all possible fragments that can be found in the entire molecular database. Obviously, this set of all existing fragments is enormous even for relatively small datasets: a single molecule of average size can already contain in the order of hundreds of thousands of different fragments. Existing methods 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 fulfill the desired criteria. Based on existing data mining algorithms for market basket analysis [2,3] these methods conduct depth-first [7] or breadth-first searches [6,5]. An example of a search tree is depicted in Fig. 1, which also shows the region of discriminative fragments.

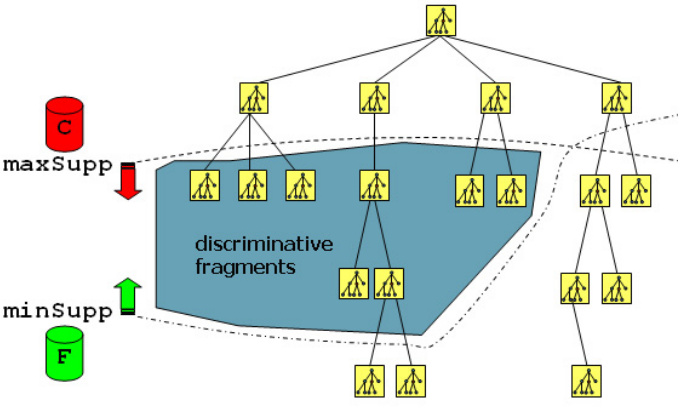

Fig. 1. Discriminative molecular fragment search tree

However, none of the current sequential 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. Although, in recent years, several parallel and distributed algorithms have been proposed for the association rule mining problem (D-ARM) [9], very few have addressed the FSM problem [10,11]. The approach in [10] achieved a relatively good performance in a small-scale computational environment, but its scalability and efficiency are limited by two main factors. First, the approach is based on a master-slave communication model, which clearly cannot scale well to a large number of computing nodes. Secondly, the communication overhead due to the large number of frequent fragments limits the efficiency of the overall process. In this paper, we overcome these two limitations by adopting a better definition of discriminative fragments and by providing a more efficient and scalable distributed computing framework.

3 Efficient Frequent Subgraph Mining

We can assume that the molecular compounds in the dataset can be classified in two groups, the focus set F (active molecules) and its complement C (non-active molecules). For example, during the High Throughput analysis, compounds are tested for a certain active behaviour and a score associated to their activity level is determined. In this case, a threshold ($thres$) on the activity value allows the classification of the molecules in the two groups.

The aim of the data mining process is to provide a list of molecular fragments that are frequent in the focus dataset and infrequent in the complement dataset. However, the high number of frequent fragments that can be found in a large dataset suggests the adoption of the closed frequent subgraphs (CFS). A closed frequent subgraph is a frequent subgraph whose support is higher than the

support of all its proper supergraphs. Given the CFS set, it is possible to directly generate all frequent subgraphs without any further access to the dataset. Moreover, the support of all frequent subgraphs is implicitly defined by the support of the closed ones. For this reason we adopt the CFS in more efficient definitions of discriminative molecular fragments.

Given a dataset D and a frequency threshold *minSupp*, the sets of frequent and closed frequent subgraphs are defined, respectively, as

$FS_D = \{\text{s} \mid \text{supp(s,D)} \geq \text{minSupp}\}$ and
$CFS_D = \{\text{s} \mid \text{supp(s,D)} \geq \text{minSupp and } \nexists\ \text{x} \in FS_D, \text{x} \supset \text{s and}$
$\text{supp(x,D)} = \text{supp(s,D)}\}$,

where s is a graph, supp(s,D) is the number of graphs in D, which are supersets of s, i.e. the support of s in D.

In our context, we have to extend the concept of the closure to the duality of active and non-active compounds. The following alternative definitions can be adopted for the discriminative fragments (DF).

Definition 1 (Constrained FS). *DF_{all} is the set of frequent subgraphs in the focus dataset constrained to infrequency in the complement dataset, according to*

$$DF_{all} = \{s \in FS_F \mid supp(s,C) \leq maxSupp\}.$$

Definition 2 (Constrained Focus-closed FS). *DF_F is the set of closed frequent subgraphs in the focus dataset constrained to infrequency in the complement dataset, according to*

$$DF_F = \{s \in CFS_F \mid supp(s,C) \leq maxSupp\}.$$

Definition 3 (Constrained Closed FS). *DF_{FC} is the set of frequent subgraphs in the focus dataset constrained to infrequency in the complement dataset, which are closed w.r.t. both sets of graphs, according to*

$$DF_{FC} = \{s \in FS_F \mid supp(s,C) \leq maxSupp$$
$$and\ \nexists\ x \in FS_F,\ x \supset s,\ supp(x,F) = supp(s,F)\ and\ supp(x,C) = supp(s,C)\}.$$

The first definition considers the subgraphs that are frequent in the focus dataset and are constrained to a maximum support in the complement dataset. In the other two definitions, the constrained frequent subgraphs are restricted by the closure, respectively, only in the focus and in both datasets.

Closed frequent substructures can be considered a compact representation of the complete set of frequent substructures and lead to a significant improvement of the efficiency of the mining process.

Table 1 provides an example of the number of discriminant fragments for the NCI HIV dataset (cf. Sect. 6) when the different definitions are adopted. It should be pointed out that the alternative definitions do not reduce the number of nodes in the search tree, but only the number of stored and reported molecular fragments.

Table 1. Discriminative molecular fragments in 37171 NCI compounds ($thres = 0.5$)

$minSupp(\%)$	DF_{all}	DF_F	DF_{FC}
20	1091	35	827
15	9890	53	2158
10	17688	120	4874
8	59270	241	6044
6	OutOfMem	441	9522

(a) $maxSupp = 1\%$

$minSupp(\%)$	DF_{all}	DF_F	DF_{FC}
20	0	0	0
15	5279	5	248
10	9403	24	977
8	50222	98	1728
6	127773	155	2629

(b) $maxSupp = 0.1\%$

3.1 Sequential Subgraph Mining

The distributed approach presented in this paper is based on the sequential algorithm described in [7]. The algorithm organizes the space of all possible fragments in an efficient depth-first search tree. Each possible subgraph of the molecular structures is evaluated in terms of the number of embeddings that are present in the molecular database. 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. 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. The algorithm prunes the DFS tree according to three criteria. The support-based pruning exploits the anti-monotone property of the fragment support. The size-based pruning exploits the anti-monotone property of the 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 [7].

The analysis of the sequential algorithm 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, nor bounded.

4 Distributed Subgraph Mining

The distributed approach we propose is based on a search space partitioning strategy, distributed task queues with dynamic load balancing and a peer-to-peer communication framework.

A static load balancing policy cannot be adopted as the work load is not known in advance and cannot be estimated. We adopted a receiver-initiated DLB approach based on two components, a quasi-random polling for the donor selection and a work splitting technique for the subtask generation. Both components contribute to the overall DLB efficiency and to its suitability to heterogeneous computing resources.

It is worth mentioning that all the algorithms, which have been proposed for D-ARM in the past years, assume a static, homogeneous and dedicated computation environment and do not provide dynamic load balancing [9].

In the next section, we discuss some details of the distributed application related to the search space partitioning.

4.1 Search Space Partitioning

Partitioning a Depth First Search (DFS) tree, i.e. parallel backtracking [12], has been widely and successfully adopted in many applications. In general, it is quite straightforward to partition the search tree to generate new independent jobs, which can be assigned to idle processors. In this case, no synchronization is required among remote jobs.

A job assignment contains the description of a search node of the donor worker, which becomes the initial fragment from which to start a new search at the receiving worker. The job assignment must contain all the information needed to continue the search from exactly the same point in the search space. In our case, this is essential in order to exploit the efficient search strategy provided by the sequential algorithm and based on advanced pruning techniques. Thus, a job description includes the search node state to rebuild the same local order necessary to prune the search tree as in the sequential algorithm (cf. structural pruning in [7]). This requires an explicit representation of the state of the donated search node. For this aim, we adopted the Simplified Molecular Input Line Entry Specification (SMILES) [13], a notation for organic structure description, which we enhanced with numerical tags for atoms. These tags are used to represent the subscripts of the atoms in a fragment according to the local order, i.e. the order in which atoms have been added to the fragment.

The enhanced-SMILES representation of the fragment plus the last extension performed (last extended atom subscript, last extended bond type and last added atom type) are sufficient to re-establish the same local order at a remote process. The receiving worker has to re-compute all the embeddings of the core fragment into all molecular compounds in order to re-start the search. This extra computation is necessary and is by far preferred over the expensive communication cost of an explicit representation of the embeddings. The number of embeddings of a fragment in the molecules can be very large, especially in the lower part of the search tree. Moreover, the donor worker can also perform a selection and a projection of the dataset based on the donated search node.

Each worker maintains only a local and partial list of substructures found during the execution of subtasks. Therefore, at the end of the search process we perform a reduction operation. Workers are organized in a communication tree and the number of communication steps required is in the order of $O(\log N)$, where N is the number of processes. However, the determination of the closed fragments includes expensive graph and subgraph isomorphism tests and may represent a non-trivial computational cost. Therefore, the selection of the closed fragments has to be distributed as well. This is performed during the reduction operation in parallel by several concurrent processes.

A static partition of the search space can be adopted when job running times can be estimated, which is not our case. We adopted a dynamic search tree partitioning with a self-adaptive job-granularity based on a quasi-randomized dynamic load balancing, which is discussed in the next section.

5 Dynamic Load Balancing (DLB)

Many DLB algorithms for irregular problems have been proposed in the literature and their properties have been studied. Most of them rely on uniform [14] or bounded [15] task times or the availability of workload estimates [16]. However, none of these assumptions holds in our case; we cannot guarantee that the computation cost of a job is greater than the relative transmitting time, nor 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.

In general, the DLB policy has to provide a mechanism to fairly distribute the load among the processors using a small number of generated subtasks to reduce the communication cost and the computational overhead. In particular, the quality of both the selection of donors and the generation of new subtasks is fundamental for an effective and efficient computational load distribution. These two tasks are carried out, respectively, by the DLB algorithm and the work splitting-mechanism discussed in the next two sections.

5.1 Quasi-Random Polling

When a worker completes its task, it has to select a donor among the other workers to get a new subtask. In general, not all workers are equally suitable as donor. Workers that are running a mining task for a longer time, have to be preferred. This choice can be motivated by two reasons. The longest running jobs are likely to be among the most complex ones. And this probability increases over time. Secondly, a long job-execution time may also depend on the heterogeneity of the processing nodes and their loads. With such a choice we provide support to the nodes that are likely overloaded either by their current mining task assignment or by other unrelated processes.

The DLB approach we adopted is a receiver-initiated algorithm based on a distributed quasi-random polling. Each worker keeps an ordered list of potential donors and performs a random polling over them to get a new task. The probability of selecting a donor from the list is not uniform. In particular, we adopt a simple linearly decreasing probability, where the donor list is ordered according to the starting time of the latest job assignment. This way, long running jobs have a high probability of being further partitioned, while most recently assigned tasks do not.

In order to maintain statistics of job executions, we adopted a centralized approach. At the starting and at the completion of a job execution, workers notify the bootstrap node, which collects global job statistics. Workers keep the local donor list updated by an explicit query to the bootstrap node.

Approaches based on global statistics are known to provide optimal load balancing performance, while randomized techniques provide high scalability.

In order to reduce latency, each worker also keeps a local pool of unprocessed jobs. This way at the completion of a job, the request and reception of a new one can be overlapped to the execution of a job from the local pool.

Furthermore, each worker keeps a list of donated and not completed jobs in order to support mechanisms for fault tolerance and termination detection.

It should be noticed that the server for job statistics plays the same role as the centralized directory of the first-generation P2P systems. The current implementation of our P2P computing framework allows the dynamic joining of peers and a basic fault-tolerance mechanism in case of abrupt peer disconnection.

5.2 Work Splitting

In problems with uniform or bounded subtask 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 tremendously. 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 a mechanism at the donor to reduce the probability of generating trivial tasks and of inducing idling periods at the donor processor itself. Search nodes from the stack can only be donated (a) if they have sufficient support in the active compounds and (b) if they do not have a very restrictive local order. A node with a restrictive local order is likely to generate a small subtree even in the case of high support.

A worker follows three rules to donate a search node from its local stack. A search tree node n can only be donated if

1. $stackSize() \geq minStackSize$,
2. $support(n) \geq (1 + \alpha) * minSupp$,
3. $lxa(n) \leq \beta * atomCount(n)$,

where α and β are tolerance factors, $lxa()$ is the subscript of the last extended atom in the fragment (see below), $atomCount()$ provides the number of atoms in a fragment and $minStackSize$ specifies a minimum number of search nodes in the stack to avoid starvation of the donor. The values of these parameters are not critical and in our experiments we adopted $minStackSize = 4$, $\alpha = 0.1$ and $\beta = 0.5$. These rules for selecting nodes of the local search tree guarantee that the worker does not run out of work while donating non-trivial parts of its search tree.

While rules 1 and 2 are quite straightforward, in order to explain rule 3, we have to refer to the structural pruning technique adopted in the sequential algorithm (cf. [7]). An atom subscript indicates the order in which the atom has been added to the fragment. All the atoms of the fragment with a subscript less

than lxa cannot be further extended according to the sequential algorithm. As a consequence, subtrees rooted at a node with a high lxa value (close to the number of atoms in the fragment) are expected to have a low branching factor.

6 Experimental Results

The distributed algorithm has 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. 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 a total of 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 the focus dataset we combined these compounds as follows. We joined the CA set with a different number of CM compounds to form four focus datasets with, respectively, 325, 650, 975 and 1200 compounds.

Experimental tests have been carried out on a network of workstations[1]. The software has been developed in Java; the communication among processes has been implemented using TCP socket API and XML data format.

In our tests, we introduced a synchronization barrier to wait for a number of processors to join the P2P system before starting the mining task only in order to collect performance results. In general, this is not necessary, but in the following results we did not want to consider the latency that is required to simply start up the remote peers.

In general, the mining task becomes more difficult when the absolute value of the minimum support decreases. In this case, a bigger and deeper part of the fragment lattice has to be explored. We fixed $minSupp = 6\%$ and varied the number of molecules in the focus dataset in order to show the influence of the different definitions of Sect. 3 on the running time. For the different focus datasets that have been defined above, this corresponds to an absolute minimum support, respectively, of 20, 39, 59, and 72 molecules.

A comparison of running times of the serial and distributed (over 8 processors) algorithms is shown in Fig. 2. The serial algorithm (serial DF_{FC}) and one parallel version (parallel DF_{FC}) search for the closed frequent fragments according to definition 3. The other two parallel versions search for all frequent fragments (parallel DF_{all}) and for the discriminative fragment of definition 2 (parallel DF_F). It is evident that mining the dataset for all frequent fragments (DF_{all}) can become quite an expensive task. The running time of parallel DF_{all}

[1] Nodes have different hardware and software configurations. The group of the eight highest performing machines is equipped with a CPU Intel Xeon 2.40GHz, 3GB RAM and run Linux 2.6.5-7.151 as well as Java SE 1.4.2_06.

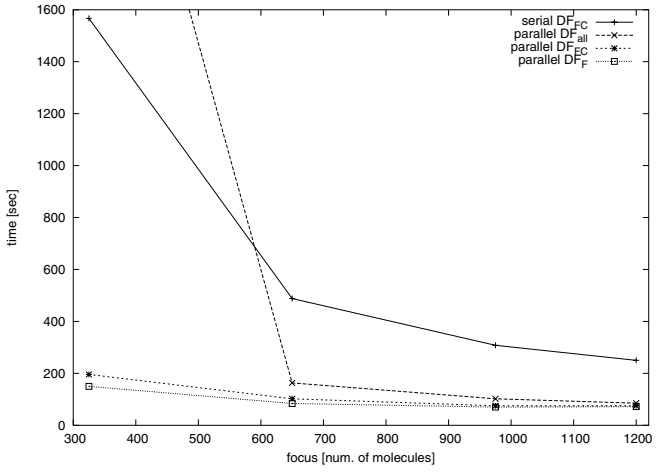

Fig. 2. Running time comparison (minSupp=6% maxSupp=1%)

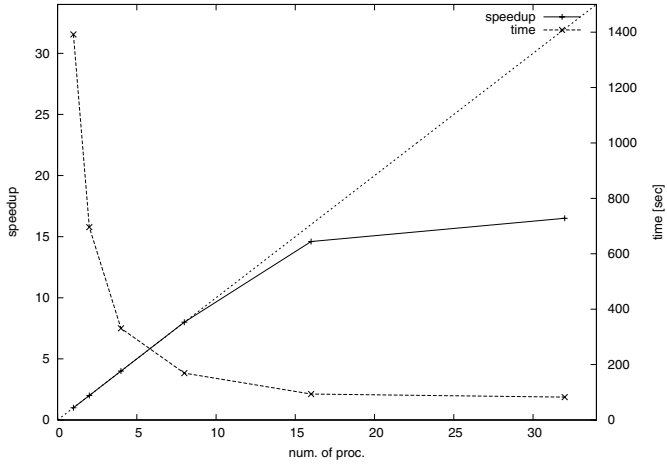

Fig. 3. Speedup and running time (minSupp=6%, maxSupp=1%)

for 325 active molecules was above 3000 seconds. This is due to the combinatorial explosion of the number of frequent fragments. It should be mentioned that, in this case (DF_{all}), the sequential algorithm cannot even complete the mining task due to the single-system memory limitations.

Mining the dataset for the closed fragments (DF_F and DF_{FC}) is feasible for the serial algorithm and is significantly sped up by the parallel execution.

We complete the analysis of the distributed approach by showing the speedup curve (Fig. 3) of the parallel over the serial algorithm, when they search for the discriminative fragments of definition 3 (DF_{FC}). The speedup is linear in the first part of the chart. Then, it is evident that more resources cannot further decrease

the running time because the amount of work is not significant enough and additional computational resources cannot be effectively exploited. Nevertheless, it is positive that the running time does not increase when unnecessary resources are used as one might expect because of the additional communication and computation overheads. This provides evidence of the good scalability properties of the system.

7 Conclusions

In this paper we presented a high performance computing approach to the frequent subgraph mining problem for the discovery of discriminative molecular fragments. The adopted approach is based on three components, which are a dynamic partitioning of the search space, a novel dynamic load balancing policy and a peer-to-peer communication framework. Very low communication and synchronization requirements, quasi-randomized receiver-initiated load balancing and high scalability of the communication framework make this distributed data mining application suitable for large-scale, non-dedicated, heterogeneous computational environments like Grids. Furthermore, the proposed approach naturally tolerates node failures and communication latency and supports dynamic resource aggregation. Experimental tests on real molecular compounds confirmed its effectiveness.

Future research effort will focus on very large-scale systems, where the centralized server for collecting job statistics could potentially become a bottleneck.

Acknowledgements

This work was supported by the Italian National Research Council (CNR) and the DFG Research Training Group GK-1042 "Explorative Analysis and Visualization of large Information Spaces".

References

1. Washio, T., Motoda, H.: State of the art of graph-based data mining. ACM SIGKDD Explorations Newsletter **5** (2003) 59–68
2. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, (Washington, D.C.)
3. Zaki, M., Parthasarathy, S., Ogihara, M., Li, W.: New algorithms for fast discovery of association rules. In: Proceedings of 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD'97). (1997) 283–296
4. Deshpande, M., Kuramochi, M., Karypis, G.: Frequent sub-structure-based approaches for classifying chemical compounds. In: Proceedings of IEEE International Conference on Data Mining (ICDM'03), Melbourne, Florida, USA (2003)
5. Deshpande, M., Kuramochi, M., Karypis, G.: Automated approaches for classifying structures. In: Proceedings of Workshop on Data Mining in Bioinformatics (BioKDD). (2002) 11–18

6. Yan, X., Han, J.: gSpan: Graph-Based Substructure Pattern Mining. In: Proc. of the IEEE Int. Conference on Data Mining, Maebashi City, Japan (2002)
7. Borgelt, C., Berthold, M.R.: Mining molecular fragments: Finding relevant substructures of molecules. In: IEEE International Conference on Data Mining (ICDM 2002), Maebashi, Japan (2002) 51–58
8. Kramer, S., de Raedt, L., Helma, C.: Molecular feature mining in hiv data. In: Proceedings of 7th Int. Conf. on Knowledge Discovery and Data Mining, (KDD'01), San Francisco, CA (2001) 136–143
9. Zaki, M.J.: Parallel and distributed association mining: A survey. IEEE Concurrency **7** (1999) 14–25
10. Di Fatta, G., Berthold, M.R.: Distributed mining of molecular fragments. In: IEEE DM-Grid Workshop of the Int. Conf. on Data Mining, Brighton, UK (2004)
11. Wang, C., Parthasarathy, S.: Parallel algorithms for mining frequent structural motifs in scientific data. In: Proceedings of the 18th Annual International Conference on Supercomputing (ICS'04), Saint Malo, France (2004)
12. Finkel, R., Manber, U.: DIB - a distributed implementation of backtracking. ACM Transactions on Programming Languages and Systems **9 (2)** (1987) 235–256
13. Daylight Chemical Information Systems, Inc.: SMILES - Simplified Molecular Input Line Entry Specification. (In: http://www.daylight.com/smiles)
14. Karp, R., Zhang, Y.: A randomized parallel branch-and-bound procedure. In: Proceedings of the 20 Annual ACM Symposium on Theory of Computing (STOC 1988). (1988) 290–300
15. Chakrabarti, S., Ranade, A., Yelick, K.: Randomized load-balancing for tree-structured computation. In: Proceedings of the Scalable High Performance Computing Conference (SHPCC '94), Knoxville, TN (1994) 666–673
16. Chung, Y., Park, J., Yoon, S.: An asynchronous algorithm for balancing unpredictable workload on distributed-memory machines. ETRI Journal **20** (1998) 346–360
17. Weislow, O., Kiser, R., Fine, D., Bader, J., Shoemaker, R., Boyd, M.: 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, UK, **81** (1989) 577–586