The 17th IASTED Int. Conf. on Parallel and Distributed Computing and Systems (PDCS 2005), Nov. 14-16, 2005, Phoenix, AZ, USA

EFFICIENT MINING OF DISCRIMINATIVE MOLECULAR FRAGMENTS

Giuseppe Di Fatta
University of Konstanz, Germany
and ICAR-CNR,
Consiglio Nazionale delle Ricerche, Italy
difatta@inf.uni-konstanz.de

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

ABSTRACT
Frequent pattern discovery in structured data is receiving an increasing attention in many application areas of sciences. However, the computational complexity and the large amount of data to be explored often make the sequential algorithms unsuitable. In this context high performance distributed computing becomes a very interesting and promising approach. In this paper we present a parallel formulation of the frequent subgraph mining problem to discover interesting patterns in molecular compounds. The application is characterized by a highly irregular tree-structured computation. No estimation is available for task workloads, which show a power-law distribution in a wide range. The proposed approach allows dynamic resource aggregation and provides fault and latency tolerance. These features make the distributed application suitable for multi-domain heterogeneous environments, such as computational Grids. The distributed application has been evaluated on the well-known National Cancer Institute's HIV-screening dataset.

KEY WORDS
Distributed computing, frequent subgraph mining, dynamic load balancing, biochemical databases.

1 Introduction

Frequent pattern discovery in structured data is receiving an increasing attention in several areas of the science. In particular, in molecular biology it is often desirable to find common properties in large numbers of drug candidates. A crucial step in the drug discovery process is the so-called High Throughput Screening and the subsequent analysis of the generated data. A promising approach focuses on the analysis of the molecular structure and the extraction of relevant molecular fragments that may be correlated with activity. Such fragments can be used to directly identify groups of promising molecules (clustering) because of their representation that is immediately understandable to chemists and biologists. They can also be used to predict activity in other compounds (classification) [1] and to guide the synthesis of new ones.
The discovery of relevant molecular fragments can be formulated as a frequent subgraph mining (FSM) problem [2] in analogy to the association rule mining (ARM) problem [3, 4]. While in ARM the main structure of the data is a list of items (itemset) and the basic operation is the subset test, FSM relies on graph and subgraph isomorphism.
Sequential algorithms are limited by single processor computing resources and are often unsuitable for extremely large datasets and an unlimited size of the fragments that can be discovered.
In this paper, we present a distributed application of the frequent subgraph mining problem for molecular compounds. We define the relevant molecular fragments in terms of frequent subgraphs and discuss the efficiency of the mining task. The distributed algorithm is based on a search space partitioning strategy with an efficient structural pruning technique. The analysis of the search space pointed out the highly irregular computation load. Several dynamic load balancing (DLB) algorithms have been proposed in the last years to allow an efficient distribution of the computation load for irregular problems. Nevertheless, some of their assumptions do not hold in this particular application. We adopt a novel DLB technique, a scheduler-based quasi-random polling policy, within a message-passing communication framework.
The distributed algorithm has been applied to the analysis of real molecular compounds, the National Cancer Institute's HIV-screening dataset.

The rest of the paper is structured as follows. In the next section we introduce the molecular fragment mining problem and present alternative definitions of discriminative molecular fragments that influence the efficiency of the overall mining process. We also discuss the irregular computation that characterizes the sequential algorithm. In section 3, we present a parallel computing approach for molecular fragment mining and the adopted DLB policy. Section 4 describes the experiments we conducted to evaluate the performance of the parallel approach. Finally, we provide conclusive remarks.

2 Mining Molecular Fragments

The problem of selecting relevant 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 an aromatic 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.
Most of the existing methods to find frequent patterns attempt to implicitly organize the search space in a lattice, which models subgraph relationships. 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 the underlying problem.
Finding frequent subgraphs in a set of graphs involves graph and subgraph isomorphism testing, which are computationally expensive. The subgraph isomorphism test is known to be an NP-complete problem [9]. Furthermore, there exists no known polynomial algorithm for isomorphism testing of general graphs, although the problem has not been shown to be NP-complete. In [10] the problem has been assigned to a special graph-isomorphism complete complexity class, which falls between the P and NP-complete classes. However, it is known that it can be solved in polynomial time for many restricted classes of graphs, such as bounded-degree graphs [11]. Molecular compounds fall in the latter case. Nevertheless, the combinatorial nature of the problem poses a great challenge. The set of all frequent 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.

The parallel approach presented in this paper is based on the sequential algorithm (MoFa) described in [7]. The algorithm visit the fragment lattice, i.e. the space of all possible subgraphs that are present in the molecular compounds, in an efficient search tree. The algorithm performs an exhaustive depth-first search and 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, which is able to restrain duplicate generation. For further details on the algorithm we refer to [7].

2.1 Discriminative fragments

We assume that the molecular compounds in the dataset can be classified in two groups. We refer to the two classes of molecules as the focus set (active molecules) and its complement (inactive 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.
Discriminative molecular fragments are contrast substructures that are frequent in a predefined subset of molecules

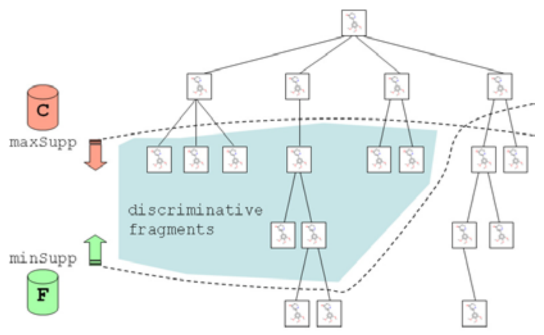

Figure 1. Discriminative molecular fragment search tree

(focus) 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. An example of a search tree is depicted in figure 1, which also shows the region of the discriminative fragments.
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 and to guide the synthesis of new ones. For example, they can be adopted as features in a multidimensional space in molecular compounds classification systems [1].
However, the high number of frequent fragments that can be found in a large dataset suggests the adoption of subsets of the frequent subgraphs for a more efficient computation. Closed frequent subgraphs (CFS) are known to provide the same topological information on the search space as the frequent ones. 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, for example, to generate all frequent subgraphs without any further access to the dataset. Moreover, the support of all frequent subgraphs is implicitly defined by the closed subgraphs. For this reason we can 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 = {s | supp(s,D) $\geq$ minSupp} and

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

where s is a graph, supp(s,D) is the number of graphs in D, which are supergraphs 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 inactive compounds. The following different definitions can be adopted for 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:*

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

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, in only the focus dataset and in both datasets.

The closed frequent substructures can lead to a significant improvement of the efficiency of the mining process. For example, only for reporting purposes if the search algorithm does not guarantee the uniqueness of the discovered fragments, the number of graph isomorphism tests, which need to be performed, is in the order of $O(n^2)$, where n is the number of frequent subgraphs. Moreover, the frequent fragments might also overwhelm the memory of a single computing node. In a distributed computational environment, where partial results of remote processes have to be collected, closed frequent fragments also lead to another significant advantage, i.e. a lower communication overhead.

Figure 2 provides an example of the number of frequent and discriminative fragments for the NCI HIV dataset (cf. section 4) when the given definitions are adopted. It is evident how fewer closed fragments can carry equivalent information to all frequent ones. For small values of the support threshold, their ratio can achieve several orders of magnitude. Moreover, in extreme cases we may not be able to keep in memory all the frequent fragments because of the limitations of a single processor. It should be noticed 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.

In general, definition 3 is preferred, because it already provides a significant reduction of the cardinality of the reported-fragment set, while it still maintains interesting information about the support in the complement dataset.

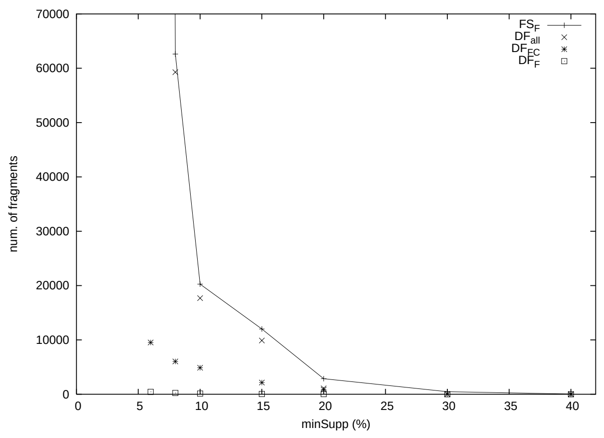

Figure 2. Number of molecular fragments

2.2 Irregular search space

The analysis of the sequential algorithm points out the irregular nature of the search tree. An irregular problem is characterized by a highly dynamic and unpredictable domain. In this application, the time complexity to visit the search tree, and even to explore a single search tree node cannot be estimated. The data mining nature of the problem makes the time required to visit a node unpredictable. The pruning techniques, which this kind of search algorithms heavily relies on, make the estimation of the tree exploration time very difficult. Depth and fan of the search tree are also unpredictable. In order to provide evidence of the above considerations, we collected statistics of the running time required by the sequential algorithm to visit the subtree rooted at each node of the entire search tree ($minSupp = 10\%$ and $maxSupp = 1\%$). Figure 3 shows that the subtree visiting time follows a power-law distribution. If we consider each subtree as a potential subtask for a distributed computational environment, there is a very large number of very small subtasks and a small number of large subtasks.

Moreover, no assumption can be made on the lower bound of subtask workloads. In general, a single node exploration can take from few milliseconds to several minutes. Thus, in the design of a parallel approach we cannot assume that subtask transmission time is less than its computation cost.

Static and most dynamic load balancing policies are unsuitable for this application. We have to adopt a strategy for task partitioning and distribution that is able to reduce the generation of trivial tasks, while can still balance the load among the available resources.

3 Parallel discriminative fragment mining

In recent years, several parallel and distributed algorithms have been proposed for the association rule mining pro-

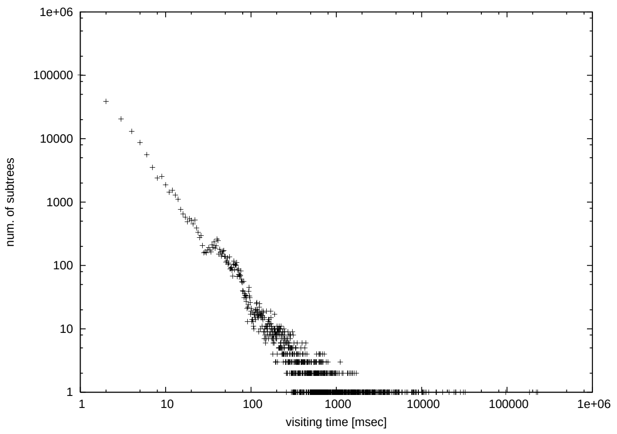

Figure 3. Distribution of the subtree visiting time

blem (D-ARM) [12]. All these algorithms assume a static, homogeneous and dedicated computation environment and do not provide dynamic load balancing.

However, currently very few parallel and distributed FSM algorithms have been proposed in the literature [13][14][15]. Although parallel search algorithms have been studied for a long time, distributed data mining applications, like FSM, are still non-trivial. The complexity of the problem and the large amount of data to be explored make parallel formulations of these methods extremely challenging, especially when the target HPC architecture is a distributed heterogeneous system.
The approach in [13] 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 work, we overcome these limitations by adopting a better definition of discriminative fragments and by providing a more efficient and scalable DLB policy.

In order to adopt a distributed approach in a large-scale computing environment, communication latency and node failures have to be tolerated. For this reason we have adopted a partitioning strategy and discarded solutions based on a collaborative approach among processes, such as distribution techniques similar to ones proposed in [16, 17], which are more suited for dedicated HPC systems.

The distributed approach we propose is based on three main aspects:

- a search space partitioning strategy,
- a centralized task queue with dynamic load balancing,
- and a message-passing communication framework.

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.
In the next sections we discuss some details of the distributed application related to the search space partitioning and to the load balancing policy.

3.1 Search space partitioning

Partitioning a Depth First Search (DFS) tree, i.e. parallel backtracking [18], 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.

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]).

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. This is more efficient than the star topology adopted in the master-slave approach of [13], which requires $O(N)$ sequential communication steps. During the reduction of the frequent fragments, partial local lists are merged and duplicates are discarded. In case only closed fragments are required, non-closed ones could be filtered out at the end of the reduction process. However, the determination and selection of the closed fragments include expensive subgraph isomorphism tests and may represent a non-trivial computational cost for a single processor. Furthermore, as shown in section 2.1, the number of all frequent fragments may become very large and the communication cost would be too expensive. Therefore, the selection of the closed fragments has been distributed as well. This is performed during the reduction operation in parallel by several concurrent processes and not only by a single master node.
Thus, the reduction operation based on a logical communication tree has the advantage of reducing the number of communication steps and distributing the computational load associated to the costly graph and subgraph isomorphism tests for the closed fragment selection.

3.2 Dynamic load balancing

Many DLB algorithms for irregular problems have been proposed in the literature and their properties have been studied [19]. Most of them rely on uniform [20] or bounded [21] task times or the availability of workload estimates [22]. 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 subsections.

3.2.1 Scheduler-based quasi-random polling

The DLB approach we adopted is a receiver-initiated algorithm based on a centralized job-pool and a quasi-random polling. When a worker completes its task, it requests a new subtask to a centralized scheduler, which maintains a pool of available jobs and has to select job-donors among the busy workers. 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 or by other unrelated processes.

The scheduler keeps an ordered list of potential donors and performs a quasi-random polling over them to get a new task. The probability of selecting a donor from the list is not uniform, as it would be in a random polling. 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 a centralized job-pool approach it is quite simple to maintain global statistics of job executions. At the starting and at the completion of a job execution, workers notify the scheduler. The complete knowledge of job statistics in the system allows the centralized scheduler to determine suitable job donors. 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.
The scheduler keeps a list of assigned and not completed jobs in order to support mechanisms for fault tolerance and termination detection.

In some aspects, our approach is similar to the one described in [19] as a "modified" scheduler-based load balancing. However, the approach described in [19] adopts a round robin donor selection among the workers. In our case, the use of a quasi-randomized technique is fundamental for the selection of the more suitable donors to avoid the generation of a high number of trivial tasks. Polls are not uniformly distributed among all workers, but they are still spread over several donors.

3.2.2 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, 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 three rules at the donor to reduce the probability of generating trivial tasks and of inducing idle periods at the donor processor itself. A worker can donate a search node n from its local stack only if:

1. $stackSize() \geq minStackSize$,
2. $support(n) \geq (1 + \alpha) * minSupp$, and
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 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.

4 Performance evaluation

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 [23]. This screen utilized a soluble formazan assay to measure protection of human CEM cells from HIV-1 infection. 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 (0, 325, 650, 875) to form four focus datasets of size $f1 = 325$, $f2 = 650$, $f3 = 975$ and $f4 = 1200$.

Experimental tests have been carried out on a network of eight 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 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 would be explored. This can be achieved by decreasing either the relative minimum support or the number of the active molecules, i.e. the focus dataset. We decided to fix $minSupp = 6\%$ and to vary the number of molecules in the focus dataset in order to show the influence of the different definitions of section 2.1 on the running time. For the different focus datasets that have been defined above ($f1, f2, f3, f4$), 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 figure 4 ($thres = 0.5$, $minSupp = 6\%$ and $maxSupp = 1\%$). 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} for $f1$ was above 3000 seconds. This is due to the combinatorial explosion of the number of frequent fragments and, in this case, the task may become prohibitive even for a parallel computational environment. 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.

[1]The computing nodes are equipped with a CPU Intel Xeon 2.40GHz, 3GB RAM and run Linux 2.6.5-7.151 and Java SE 1.4.2_06.

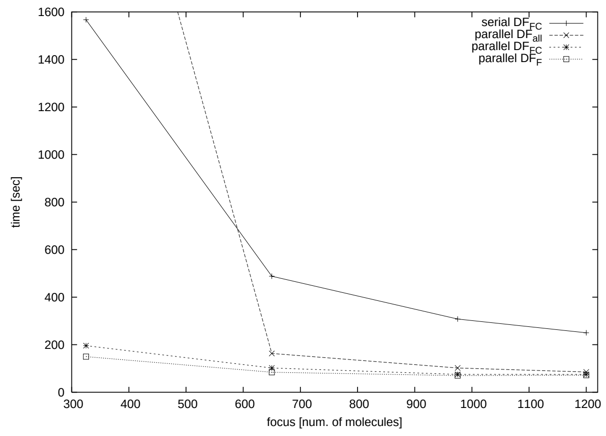

Figure 4. Running time comparison

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.

5 Conclusions

In this paper we presented a parallel approach to the frequent subgraph mining problem. The distributed algorithm has been applied to the task of molecular fragment discovery to study the effect of different discriminative fragment definitions. Several issues have also been discussed for an effective design of a large-scale distributed FSM algorithm. The adopted approach is based on three components, which are a partitioning criterion of the search space, a dynamic load balancing policy and a message-passing communication architecture. Very low communication and synchronization requirements, and good scalability of the quasi-randomized load balancing make this distributed data mining application suitable for multi-domain, heterogeneous computational environments like Computational Grids. Moreover, the proposed approach naturally tolerates node failures and communication latency and supports dynamic resource aggregation. Experimental tests on real molecular compounds in a distributed non-dedicated computing environment confirmed its effectiveness in terms of running time performance, low parallel overhead, load balancing, fault and latency tolerance.
Future research effort will focus on large-scale systems, where the centralized scheduler could potentially become a bottleneck. In this case, the DLB framework needs to be improved with distributed job-pools.

Acknowledgments

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”. We also thank the Department of Computer and Information Science of the University of Konstanz for the use of their machines.

References

[1] M. Deshpande, M. Kuramochi, and G. Karypis, "Frequent sub-structure-based approaches for classifying chemical compounds," in *Proceedings of IEEE International Conference on Data Mining (ICDM'03)*, Melbourne, Florida, USA, Nov. 19–22, 2003.

[2] T. Washio and H. Motoda, "State of the art of graph-based data mining," *ACM SIGKDD Explorations Newsletter*, vol. 5, no. 1, pp. 59–68, July 2003.

[3] R. Agrawal, T. Imielinski, and A. N. Swami, "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., May 26–28,.

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

[5] M. Deshpande, M. Kuramochi, and G. Karypis, "Automated approaches for classifying structures," in *Proceedings of Workshop on Data Mining in Bioinformatics (BioKDD)*, 2002, pp. 11–18.

[6] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *Proceedings of the IEEE International Conference on Data Mining (ICDM'02)*, Maebashi City, Japan, 2002.

[7] C. Borgelt and M. R. Berthold, "Mining molecular fragments: Finding relevant substructures of molecules," in *IEEE International Conference on Data Mining (ICDM 2002)*, Maebashi, Japan, Dec. 9–12, 2002, pp. 51–58.

[8] S. Kramer, L. de Raedt, and C. Helma, "Molecular feature mining in hiv data," in *Proceedings of 7th Int. Conf. on Knowledge Discovery and Data Mining, (KDD'01)*, San Francisco, CA, 2001, pp. 136–143.

[9] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman, 1979.

[10] S. Skiena, *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Addison-Wesley, 1990.

[11] E. M. Luks, "Isomorphism of graphs of bounded valence can be tested in polynomial time," *Journal of Computer and System Sciences*, vol. 25, pp. 42–65, Aug. 1982.

[12] M. J. Zaki, "Parallel and distributed association mining: A survey," *IEEE Concurrency*, vol. 7, no. 4, pp. 14–25, 1999.

[13] G. Di Fatta and M. R. Berthold, "Distributed mining of molecular fragments," in *IEEE DM-Grid Workshop of the Int. Conf. on Data Mining (ICDM 2004)*, Brighton, UK, Nov. 1–4, 2004.

[14] C. Wang and S. Parthasarathy, "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, June 26 - July 01, 2004.

[15] F. Schreiber and H. Schwbbermeyer, "Towards motif detection in networks: Frequency concepts and flexible search," in *Proceedings of the International Workshop on Network Tools and Applications in Biology (NETTAB04)*, Camerino, Italy, Sept. 5–7, 2004, pp. 91–102.

[16] R. Agrawal and J. Shafer, "Parallel mining of association rules," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 962–969, Dec. 1996.

[17] E. Han, G. Karypis, and V. Kumar, "Scalable parallel data mining for association rules," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 3, pp. 337–352, May/June 2000.

[18] R. Finkel and U. Manber, "Dib - a distributed implementation of backtracking," *ACM Transactions on Programming Languages and Systems*, vol. 9 (2), pp. 235–256, Apr. 1987.

[19] V. Kumar, A. Grama, and V. N. Rao, "Scalable load balancing techniques for parallel computer," *Journal of Parallel and Distributed Computing*, vol. 22, no. 1, pp. 60–79, July 1994.

[20] R. Karp and Y. Zhang, "A randomized parallel branch-and-bound procedure," in *Proceedings of the 20 Annual ACM Symposium on Theory of Computing (STOC 1988)*, 1988, pp. 290–300.

[21] S. Chakrabarti, A. Ranade, and K. Yelick, "Randomized load-balancing for tree-structured computation," in *Proceedings of the Scalable High Performance Computing Conference (SHPCC '94)*, Knoxville, TN, May 23–25, 1994, pp. 666–673.

[22] Y. Chung, J. Park, and S. Yoon, "An asynchronous algorithm for balancing unpredictable workload on distributed-memory machines," *ETRI Journal*, vol. 20, no. 4, pp. 346–360, Dec. 1998.

[23] National Cancer Institute. DTP AIDS antiviral screen dataset. [Online]. Available: http://dtp.nci.nih.gov/docs/aids/aids/data.html