Dynamic Load Balancing for the Distributed Mining of Molecular Structures

Giuseppe Di Fatta, *Member*, *IEEE*, and Michael R. Berthold, *Senior Member*, *IEEE*

Abstract—In molecular biology, it is often desirable to find common properties in large numbers of drug candidates. One family of methods stems from the data mining community, where algorithms to find frequent graphs have received increasing attention over the past years. However, the computational complexity of the underlying problem and the large amount of data to be explored essentially render sequential algorithms useless. In this paper, we present a distributed approach to the frequent subgraph mining problem to discover interesting patterns in molecular compounds. This 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 data set, where we were able to show close-to linear speedup in a network of workstations. The proposed approach also allows for dynamic resource aggregation in a nondedicated computational environment. These features make it suitable for large-scale, multidomain, heterogeneous environments, such as computational grids.

Index Terms—Distributed computing, peer-to-peer computing, dynamic load balancing, subgraph mining, frequent patterns, biochemical databases, molecular compounds.

1 Introduction

Computational biology attempts to simulate and understand the behavior of organisms often using highly complicated models. Some of these approaches target high-level interactions between cells and proteins, often referred to as "systems biology." Other approaches attempt to understand relations between gene expressions and various diseases. On an even smaller scale, molecular data analysis attempts to find common patterns that are responsible for the success of a specific medication to fight a disease. A crucial step in this 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 compounds have been automatically screened, it is necessary to select a small subset of 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. 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.

Relevant molecular fragment discovery 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 data sets and an unlimited size of the fragments that can be discovered. Sequential algorithms cannot provide scalability in terms of data size and dimensionality, nor better quality of the results (wider range for user parameters), nor dramatically shorter running times. 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.

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 main algorithm is based on a Depth-First Search (DFS) strategy (backtracking) and the distributed approach is based on a search space partitioning technique. Methods based on the DFS strategy have the advantage of requiring less memory than the breadth-first search counterparts. Most of the problems solved by search strategies are computationally intensive and parallel approaches have been largely studied, e.g., parallel backtracking [5].

The analysis of the search space pointed out the highly irregular computation load. Several dynamic load balancing algorithms have been proposed in the last years to allow an efficient distribution of the computation load for such

• *The authors are with the Department of Computer and Information Science, University of Konstanz, Box M712, 78457 Konstanz, Germany. G. Di Fatta is also with the High Performance Computing and Networking Institute of the Italian National Research Council (ICAR-CNR), Palermo, Italy. E-mail: {fatta, berthold}@inf.uni-konstanz.de.*

Manuscript received 1 July 2005; revised 13 Feb. 2006; accepted 8 Mar. 2006; published online 26 June 2006.
Recommended for acceptance by N. Amato, S. Aluru, and D. Bader.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDSSI-0311-0705.

irregular problems. Nevertheless, some of their assumptions do not hold in this particular application. We adopt a new load balancing technique based on a receiver-initiated policy within a Peer-to-Peer (P2P) communication framework.

The distributed algorithm has been applied to the analysis of real molecular compounds, the National Cancer Institute's HIV-screening data set.

The rest of the paper is structured as follows: In the next section, we introduce the molecular fragment mining problem and discuss related approaches. We discuss the definition of discriminative molecular fragments that influence the efficiency of the overall mining process. We briefly describe the sequential algorithm on which our distributed approach is based and discuss the irregular computation that characterizes it. In Section 3, we present a high performance distributed computing approach for molecular fragment mining and the adopted dynamic load balancing policy. Section 5 describes the experiments we conducted to evaluate the performance of the distributed approach. Finally, we provide conclusive remarks.

2 Mining Molecular Fragments

The problem of discovering 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. 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 that are frequent in a predefined subset of molecules (focus) and infrequent in the complement of this subset. In this case, two parameters are required: a minimum support ($minSupp$) in the focus subset and a maximum support ($maxSupp$) in the complement.

A number of approaches to find frequent molecular fragments have recently been published [6], [7], [8], but they are all limited by the complexity of the underlying problem. Some of these algorithms can, therefore, operate on very large molecular databases, but only find small fragments [6], [7], whereas others can find larger fragments, but are limited by the maximum number of molecules they can analyze [8], [9].

Finding frequent subgraphs in a set of graphs involves graph and subgraph isomorphism testing, which is computationally expensive. The subgraph isomorphism test is known to be an NP-complete problem [10]. Furthermore, there exists no known polynomial algorithm for isomorphism testing of general graphs, although the problem has not been shown to be NP-complete. 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. 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 data sets: A single molecule of average size can already contain in the order of hundreds of thousands of different fragments.

Existing methods to find frequent fragments attempt to implicitly organize the space of all possible fragments in a lattice that models subgraph relationships, i.e., edges connect fragments that differ by exactly one atom and/or bond. An example of a complete lattice induced by six molecules[1] is shown in Fig. 1. 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 [3], [4], these methods conduct depth-first [7] or breadth-first searches [6]. An example of a search tree is depicted in Fig. 2, which also shows the region of discriminative fragments.

None of these algorithms in a single processor can be used for extremely large data sets (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. A data-parallel strategy can be adopted to cope with larger data sets. However, only a task-parallel strategy, based on the partitioning of the search space can effectively cope with low minimum support values and a large maximum size of fragments. The latter is the approach presented in this paper.

In recent years, several parallel and distributed algorithms have been proposed for the association rule mining problem (D-ARM) [12]. However, currently very few parallel and distributed FSM algorithms have been presented in the literature [13], [14], [15], [16]. Although parallel search algorithms have been studied for a long time, distributed data mining applications, like FSM, are still nontrivial. 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 High Performance Computing (HPC) architecture is a distributed large-scale system.

We have developed a high performance distributed approach for the frequent subgraph mining (FSM) problem to discover interesting patterns in molecular compounds. The parallel algorithm is based on a sequential algorithm [8], which adopts a Depth First Search (DFS) strategy. The distributed approach is based on three main components, namely a dynamic partitioning of the search space [13], a distribution process based on a peer-to-peer communication framework [16], and a receiver-initiated, load balancing algorithm, which is discussed and evaluated in the present work.

The distributed FSM approach proposed in [13] pointed out that the main difficulties of a distributed backtracking

1. These molecules are made up for a descriptive purpose and do not have any real meaning.

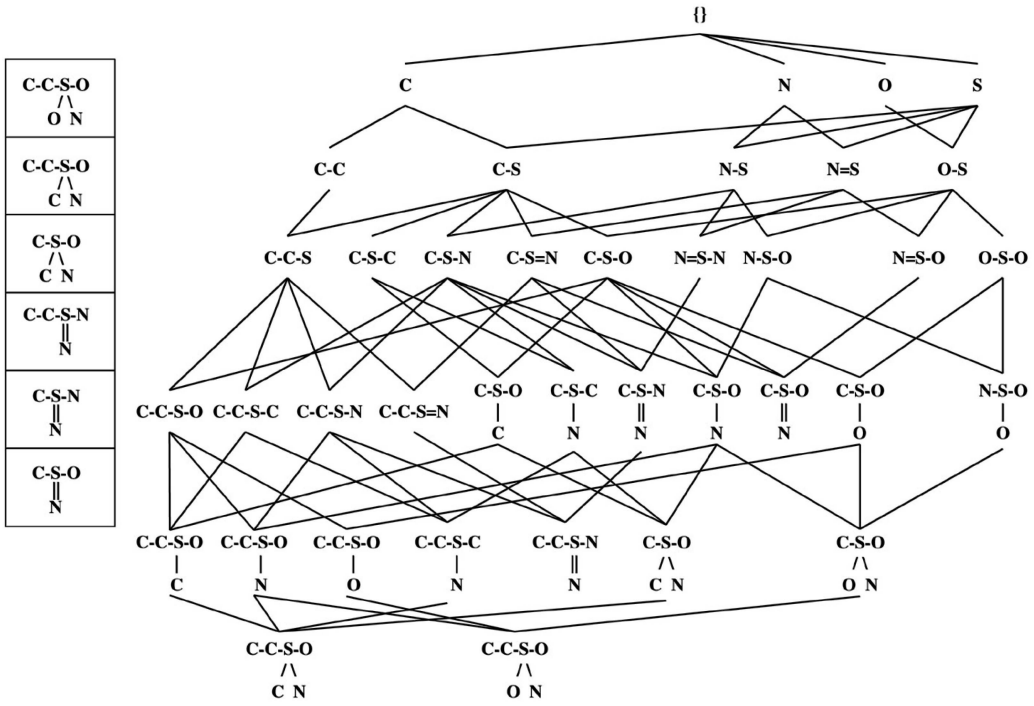

Fig. 1. Complete lattice of fragments for the six example molecules.

algorithm applied to a data mining task often arise from the highly irregular nature of the search tree. In such a case, only a Dynamic Load Balancing (DLB) policy can be successfully adopted. 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. Second, 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 [17] and an appropriate dynamic load balancing policy. As a consequence, the distributed approach can be effectively adopted in a large-scale heterogeneous computing environment, such as computational grids.

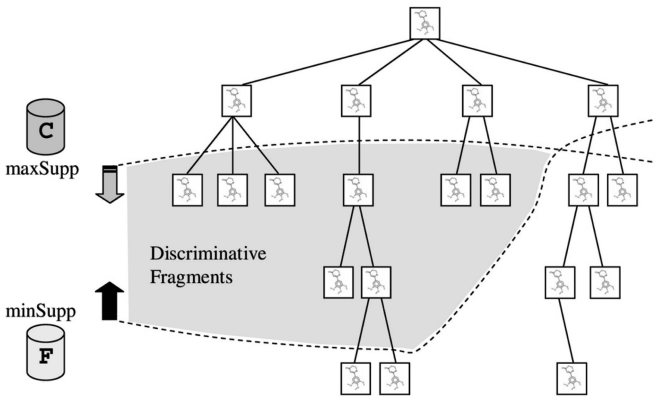

Fig. 2. Molecular fragment search tree.

In the next two sections, we introduce a compact representation of the frequent subgraphs and an efficient sequential approach to visit the entire search space.

2.1 Compact Frequent Subgraph Representations

Mining relevant molecular fragments is considered a promising tool to help the molecular biologists to identify drug candidates. In this context, we can assume that the molecular compounds in the data set can be classified in two groups. We refer to the two classes of molecules as the focus set (active molecules) and its complement (nonactive molecules). For example, during the high throughput analysis, compounds are tested for a certain active behavior 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 data set F and infrequent in the complement data set C. These topological fragments carry important information and may be representative of those components in the compounds that are responsible for a positive behavior. Such discriminative 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 multi-dimensional space in molecular compounds classification systems [1].

However, the high number of frequent fragments that can be found in a large data set 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 greater 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 data set. Moreover, the support of all frequent subgraphs is implicitly defined by the closed subgraphs. For this reason we adopt the CFS in more efficient definitions of discriminative molecular fragments.

For two graphs g and g', let $g \supset g'$ denote that g' is isomorphic to a proper subgraph of g. Given a set of graphs D and a frequency threshold $minSupp$, the sets of frequent and closed frequent subgraphs are defined, respectively, as

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

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 nonactive 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 data set constrained to infrequency in the complement data set, according to*

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

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

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

Definition 1 considers the subgraphs that are frequent in the focus data set and are constrained to a maximum support in the complement data set. In Definition 2, the constrained frequent subgraphs are restricted by the closure in the focus data set.

The concept of closed frequent substructures has been successfully adopted in both ARM and FSM approaches, in order to improve the mining process. They can be considered a compact representation of the complete set of frequent substructures. This may 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 to discard duplicates, 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 computation node. In a distributed computational environment, closed frequent fragments also lead to another advantage, a lower communication overhead.

Fig. 3 shows the number of discriminant fragments for the NCI HIV data set (cf. Section 4) when the different definitions are adopted. It is evident how fewer closed fragments can carry information that is equivalent to all frequent ones. For small values of the minimum support, their ratio can achieve several orders of magnitude. Moreover, in extreme cases we may not be able to store all the frequent fragments because of memory limitations. 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.

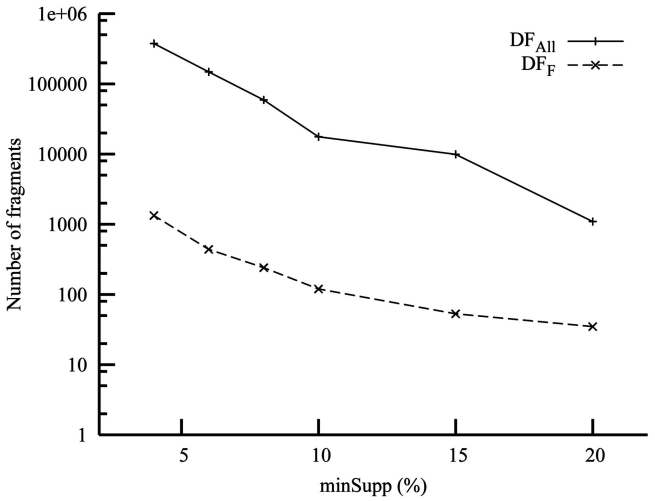

Fig. 3. Discriminative molecular fragments in 37,171 NCI compounds ($thres = 0.5$ and maxSupp = 1 percent).

From the application perspective, reporting a large number of fragments also poses a visualization and exploration problem. Thus, the reduced number of reported molecular fragments is, indeed, an advantage, provided that there is no loss of information. Among all frequent fragments, the closed ones provide a significant reduction of the cardinality, while they still maintain interesting information about the support in the active molecules. A nonclosed frequent fragment does not tell more about the active molecules than its smallest closed supergraph.

2.2 Sequential Subgraph Mining

The distributed approach presented in this paper is based on the sequential algorithm (MoFa) described in [8]. The algorithm organizes the space of all possible fragments (subgraphs) of the active compounds, i.e., the fragment lattice, in an efficient search tree. Each possible subgraph is evaluated in terms of the number of embeddings that are present in the molecular structures of both active and inactive compounds.

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 active 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 the active molecules and maximum support in the inactive molecules, it can be considered a discriminative fragment according to the adopted definition.

The algorithm finding the frequent and/or discriminative fragments employed in this paper is based on a search tree traversal, very similar to itemset and association rule

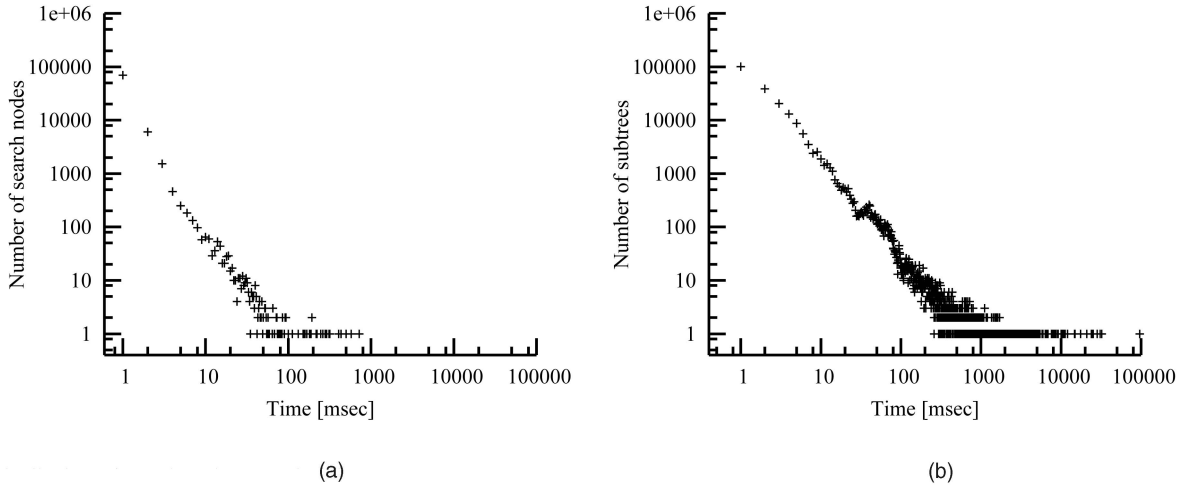

Fig. 4. Distribution of search-node expansion time and subtree visiting time ($minSupp = 10$ percent and $maxSupp = 1$ percent). (a) Search-node expansion time and (b) search subtree visiting time.

mining. In sharp contrast to those applications, it is not possible to apply a global ordering and use a prefix-tree structure to avoid duplicate combinations of items. In the case of molecules our items are atoms (or bonds and atoms) and, since any one atom type can occur multiple times at arbitrary positions in a fragment—we cannot simply first build up all carbon atoms, followed by all sulfurs and so on. The MoFa algorithm uses a sophisticated, local ordering instead. When expanding a fragment using new atoms it only considers very specific extensions at specific points. In particular, it will never consider extending an atom that has been inserted before the atom that was last extended and it will also only consider extensions that follow (using a local order on atoms and bonds) previous extensions at this atom. These two rules form the basis for the so-called "structural pruning," which, very efficiently, eliminates generation of almost all duplicate fragments. More sophisticated structural pruning methods are available as well, based on canonical forms, but we did not consider them here. These local orders have an impact on our distribution problem. Since we need to recreate those local ordering schemas on other nodes when transmitting a subtask, we also need to submit information about previously added atoms and last extensions in addition to the structure of the fragment itself. Two other pruning methods can be taken from the association rule mining algorithms more directly: support and size-based pruning. The former eliminates all fragments that do not occur often enough in our database (this criterion being monotone allows us to ignore all subsequent branches) and the latter eliminates fragments having more than a specific number of atoms. For more details on the algorithm, please refer to [8]. For the discussion here, it is sufficient to note that we can transmit the state of a node in our search tree by transferring the current fragment, the order in which the atoms were inserted and the last extension that was applied. From this information, we can regenerate the entire status information needed to continue the search.

An analysis of the sequential algorithm points 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 a few milliseconds to several minutes. It is known that the pruning techniques, which these types of search algorithms heavily rely on, make the estimation of the tree exploration time very difficult. Depth and fan of the search tree are also unpredictable. Moreover, no assumption can be made on the lower bound of subtask workloads. We cannot assume that subtask transmission time is less than the computation cost associated. This makes most dynamic load balancing policies unsuitable for this application and we have to adopt a policy that is able to reduce the generation of trivial tasks.

In order to provide evidence of the above considerations, we collected statistics of the running time required by the sequential algorithm for the expansion of each search tree node (Fig. 4a) and for the complete visit of the associated subtree (Fig. 4b). Both are characterized by a power-law distribution. This may not come as a surprise with regard to the subtree visiting time, since it is well known that this kind of distribution is typical for aggregations of multiscale hierarchies such as trees. However, it is interesting that the node expansion time shows the same type of distribution. This can be tentatively explained by the structure of the input data and the node expansion step in the sequential algorithm. Each search tree node represents a molecular fragment, which is extended in all possible successors by adding a bond and, eventually, an atom. Intuitively, this produces a high number of fast fragment extensions due to the local order and the extension rules, and a small number of computationally long extensions. However, this would require further analysis, which is out of the scope of this work.

Sequential algorithms cannot provide scalability in terms of data size and dimensionality, nor better quality of the results (wider range for user parameters), nor dramatically shorter running times. Quite obviously, parallel and distributed approaches are a promising alternative to the current sequential algorithms.

3 Distributed Subgraph Mining

Our ultimate target architecture is a large-scale multidomain computational environment, like a grid infrastructure. So, our general approach is to partition the problem into independent subtasks, in order to minimize communication and synchronization among processors. Independence among tasks would also allow the introduction of fault tolerance mechanisms more easily than in distributed applications with a complex communication pattern. 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 the ones proposed in [18], [19], which are more suited for dedicated HPC systems.

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

- 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 ranked-random polling for the donor selection and a heuristic work splitting technique for the subtask generation. Both components contribute to the overall DLB efficiency and to its suitability to heterogeneous computing resources.

A P2P communication framework naturally fits with the receiver-initiated DLB approach and provides good scalability properties. Each peer has to register at a centralized bootstrap node to join the system and to contribute to the overall computation task. Peers can directly exchange system information and computation subtasks, since each process implements both client and server functionality. The bootstrap node provides initial configuration information and a dynamic peer directory service.

Although the P2P system allows a dynamic aggregation of resources, in our tests, we introduced a synchronization barrier just for performance evaluation. When a given minimum number of peers have joined the system, the bootstrap node assigns the first job, i.e., the root node of the search tree, in order to start the distributed application.

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 [12].

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

3.1 Search Space Partitioning

Partitioning a depth-first search tree has been widely and successfully adopted in many parallel 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 (core) 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 [8]). 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) [20], 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 reestablish the same local order at a remote process. The receiving worker has to recompute all the embeddings of the core fragment into all molecular compounds in order to restart 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.

Furthermore, the donor worker can also perform a selection and a projection of the data set based on the pruned fragment. In the selection operation, the supporting subset of molecules is included in the job description. In the projection operation, each supporting molecule is shrunk to the essential subgraph, which is relevant for the subtask. The atoms and bonds in the molecule that cannot be used for further extension of the pruned fragment are not considered. The projected molecule can also be represented by means of the enhanced-SMILES representation. Data set selection and projection are adopted only when they represent a computational advantage in comparison to the disadvantage of the extra communication. This typically occurs in a later stage of the search, where larger fragments are supported by fewer molecules.

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 duplicate and nonclosed fragments are discarded.

However, the determination and selection of the closed fragments include expensive graph and subgraph isomorphism tests and may represent a nontrivial 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 nonclosed fragments has to be distributed as well. This can be performed during the reduction operation in parallel by several concurrent processes and not only by a single master node.

The reduction operation based on a logical communication tree has the advantage of reducing the number of communication steps. Moreover, it also has the advantage of distributing the computational load associated to the costly graph and subgraph isomorphism tests for the reduction of the final list of closed frequent fragments.

In our distributed application, we adopted a search-tree partitioning with a self-adaptive job-granularity based on a decentralized dynamic load balancing, which is discussed in the next section.

3.2 Dynamic Load Balancing

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 policy.

The dynamic load balancing requirement of this application is related to the same requirement of several other irregular problems based on a search tree, e.g., problems solved using the divide and conquer strategy. Many DLB algorithms for irregular problems have been proposed in the literature and their properties have been studied. However, most of them rely on assumptions on the problem, which do not hold in our case. Some DLB techniques for irregular problems are based either on the assumption of uniform or bounded task times, or on the availability of workload estimates. In [21], uniform time tasks are assumed. In [22], it is assumed that the smallest task time is comparable to or greater than the network communication time for a task. In [23], the computation, first, is evenly partitioned among processors and, successively, task migration is adopted to maintain load balance in the system. In [24], several DLB algorithms are analyzed and their scalability properties are provided in terms of the isoefficiency analysis. This study particularly addresses irregular problems where it is not possible to estimate the size of work at processors. Nevertheless, the assumptions in [24] include some characteristics of the parallel applications that in the case of subgraph mining for molecular compounds do not hold. In particular, in our application we cannot guarantee that the computation cost of a job is greater than the relative transmitting time. In general, we cannot provide a work-splitting mechanism of a good quality, as briefly discussed.

A work at a processor can be partitioned by simply removing one or more search nodes from the local stack. Each search node state has to be converted into an external and compact representation. In particular, a search node state defines a molecular fragment and the complete list of its embeddings into the supporting molecules. Such a list can be very long and cannot be conveniently converted into an external representation. Thus, the embedding list is not included in the external representation and has to be reconstructed at the receiving worker, resulting in extra computational effort (parallel overhead). The conversion from external representation to the internal one for fragments with high support values can be relatively expensive. Also, for this reason, it is important that the number of donated search nodes is kept small. For example, we cannot simply split the stack in half to generate nontrivial subtasks. Thus, we have to adopt subtasks based on single search nodes. As a consequence, according to the power-law distribution of the subtree visiting time shown in Section 2.2, it is evident that the quality of our splitting mechanism is not uniform and can be very poor. In general, we cannot even provide minimum and maximum bounds for the running time of subtasks, as we cannot provide them for the overall mining task.

It is quite challenging to efficiently parallelize irregular problems with such an unpredictable workload. Moreover, when the target computational infrastructure is a large-scale, multidomain, nondedicated environment, we cannot assume small and bounded task transmission times. In such a case, we also have to deal with the heterogeneous and dynamic load of processors and networks.

As we discussed above, in this application some assumptions adopted in common DLB approaches do not hold and these techniques are not expected to perform well. 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, in this specific application the DLB policy has to carefully select suitable subtask donors among all the workers and nontrivial subtasks among the search nodes in the local stack of a donor. 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.

3.2.1 *Ranked-Random Polling (RRP)*

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 donors. 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. 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 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 decentralized random polling with a nonuniform probability. 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 local list is not uniform. In particular, we adopt a probability that is linearly decreasing in the rank of the donor, where the list is ordered according to the starting time of the latest job assignment (rank). 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 better scalability. The RRP policy try to combine these two advantages. The centralized server at the bootstrap node allows the determination of suitable job donors from the complete knowledge of the job statistics in the system. However, this process is decoupled from the actual polling activity, which is completely decentralized. For large number of peers, the centralized server may become a bottleneck and the local information at peers out of date. Nevertheless, the query-response process for job distribution would not be delayed; only the heuristic choice of a suitable donor may get less effective. Ultimately, for very large systems, the RRP policy would be equivalent to a plain random polling.

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. Overlapping computation with communication of job request/assignment helps to avoid, or at least to reduce, most of the communication overheads. Only the latency of the first job assignment and the last job-completed message cannot be avoided at all. (In our tests, we adopted a single job buffer at each worker.)

Furthermore, each worker keeps a list of donated and not completed jobs in order to support mechanisms for termination detection and to deal with abrupt peer disconnection.

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.

In order to evaluate our DLB algorithm, we implemented a Random Polling (RP) policy with distributed job queues and a centralized job-pool approach, i.e., a Master-Slave (MS) architecture. The latter approach has been adopted in [13], where the master process always selects the worker with the longest running job as donor, in order to feed idle workers. In the proposed approach, we adopted a P2P communication framework to avoid the centralized job pool and a random-like policy with a nonuniform probability to improve the scalability of the DLB algorithm.

In some aspects, our approach is similar to the one described in [24] as a "modified" scheduler-based load balancing. In the latter, donors directly assign subtasks to idle workers, as in our case. However, in the approach described in [24], the poll is always generated by the centralized server, which adopts a round robin donor selection among the workers. In our case, the generation of job requests (polls) has also been decentralized. Moreover, in our DLB approach, the use of a heuristic technique (donor rank) is fundamental for the selection of the most suitable donors to avoid the generation of a high number of trivial tasks. In contrast to random polling, polls are not uniformly distributed among all workers. Nevertheless, they are still spread over several donors according to a stochastic process.

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, 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 idle periods at the donor processor itself. A general principle is that the donor must still have some work to do after pruning its search tree. Job donation must not cause donor starvation. This can be easily accomplished by setting a minimum stack size, which enables job donation.

A second principle is that the generation of trivial tasks reduces the efficiency and should be avoided. Donated search nodes have to be selected from the lower part of the backtracking stack, where it is more likely to find branches of the search tree with greater complexity. Fragments with a low support in the active molecules are likely to generate small subtrees and should not be donated. However, even fragments with high support can generate very small subtrees because of the restriction in the local order (cf. Section 2.2). The last condition can be heuristically estimated as described below.

In short, 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. $support(n) \geq (1 + \alpha) * minSupp$, 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 for pruning the search tree guarantee that the worker does not run out of work while donating nontrivial parts of its search tree.

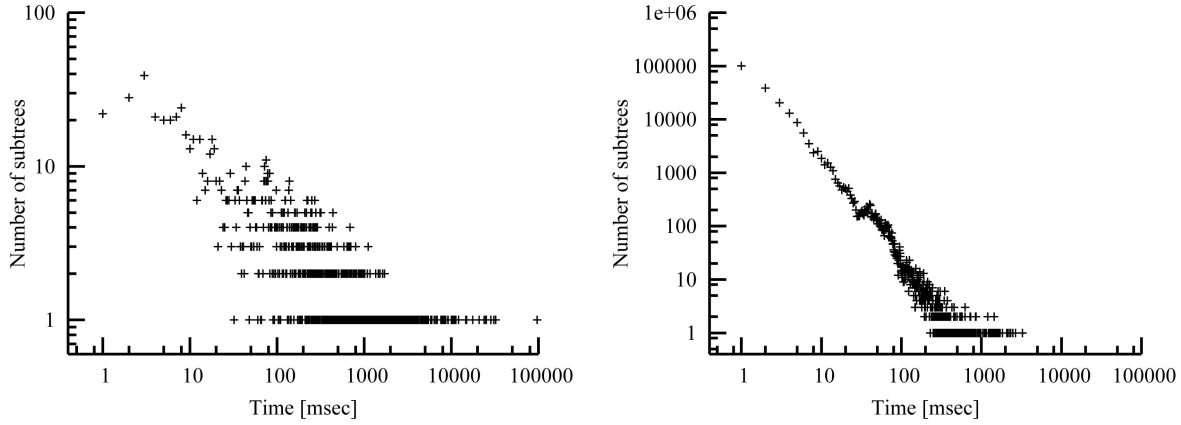

Fig. 5. Distribution of subtree visiting time with node selection rules ($minSupp = 10$ percent and $maxSupp = 1$ percent). (a) Donable nodes and (b) nondonable nodes.

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. [8]). 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.

This set of rules filters out many potentially trivial nodes (nondonable) and changes the subtree visiting time distribution of Fig. 4b to the one of Fig. 5a. The number of very small subtasks has been significantly reduced, even if they have not been completely discarded by the rules. Fig. 5b shows the distribution of the visiting times of nodes that are excluded for job donations according to the rules. In this case, the number of trivial subtasks is very large, with a distribution still similar to the original one (Fig. 4b). The moments of the time distributions in Fig. 5 are provided in Table 1, which facilitates a comparison. The great difference in the average values (μ) confirms that the selected nodes, on average, can generate more complex subtasks. However, the standard deviation value (σ) is still quite large, which means that trivial subtasks cannot be completely avoided.

The proposed DLB technique (RRP) further restricts the possible choices for new subtasks by selecting the most promising donors. This, once more, changes the distribution of the subtask visiting time. The final actual distribution is not available because it is the result of a parallel execution where subtasks are dynamically generated; task donation alters the task (subtree) at the donor itself.

TABLE 1
Subtree Visiting Time Statistics [$msec$]

max	228994	228994	3275
μ	12.5	1053	5.14
σ	740	8761	30.3

4 Performance Evaluation

In order to evaluate the proposed DLB technique (RRP), we implemented and tested a random polling algorithm (RP) and a master-slave (MS) approach. The latter, which is described in [13], is based on a centralized job-pool and new jobs are always generated by partitioning the longest running job. These three techniques use the same heuristic work-splitting mechanism described in Section 2.2.2.

We also considered a random polling policy with a simple work splitting mechanism, which is not based on heuristic techniques, nor on the application domain knowledge. Among general techniques [25], there are half splitting, taking nodes near the bottom of the stack, near a cutoff depth, or between a cutoff depth and the bottom of the stack. Half splitting would have a high cost in terms of computational overheads; each donated fragment has to be embedded in the molecules at the receiving worker. Transmitting embeddings would not be effective in terms of communication overheads, especially for large-scale systems in shared wide-area networks.

The choice of a cutoff depth would require heuristic considerations and would, anyway, depend on the particular data set and the size of molecules.

Hence, for our comparative tests, we have adopted a plain random polling, where donors take search nodes from the bottom of the local stack. This method is denoted in the following as RP_1.

All DLB methods are adopted in the same distributed application to search for the discriminative fragments of Definition 2 (DF_F), i.e., the frequent fragments that are closed in the focus data sets and constrained to infrequency in the complement data set (maxSupp = 1 percent).

4.1 Experimental Setup

The distributed algorithm has been tested for the analysis of a set of real molecular compounds—a well-known, publicly available data set from the National Cancer Institute, the DTP AIDS Antiviral Screen data set [26]. This screen utilized a soluble formazan assay to measure protection of human CEM cells from HIV-1 infection [27]. Compounds able to provide at least 50 percent protection to the CEM

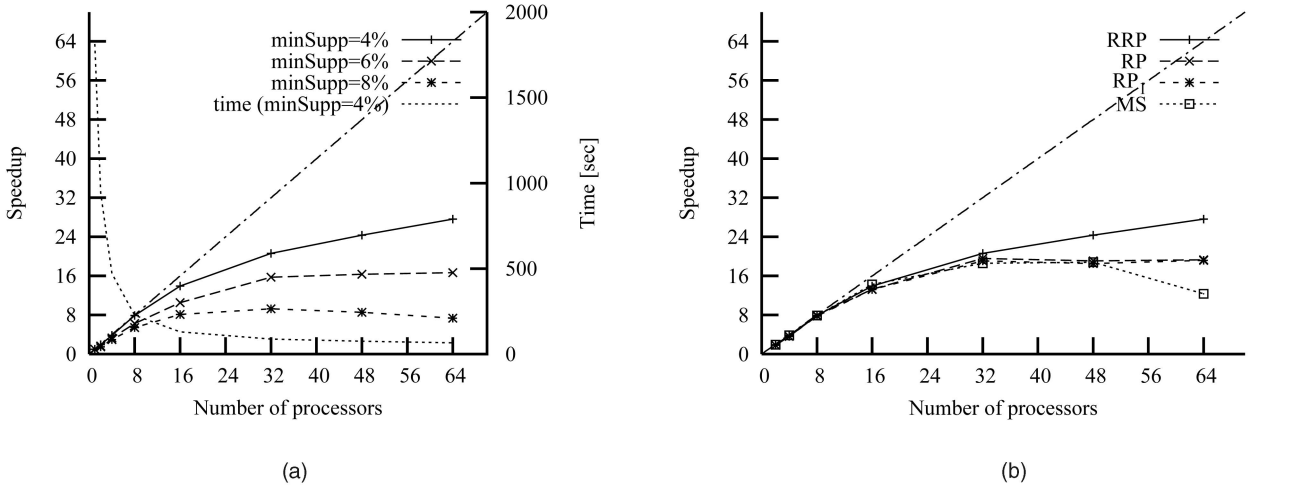

Fig. 6. Speedup and running time curves. (a) The Ranked-Random Polling (RRP) policy and (b) comparison of various DLB methods (minSupp = 4 percent).

cells were retested. Compounds that provided at least 50 percent protection on retest were listed as moderately active (CM). Compounds that reproducibly provided 100 percent protection were listed as confirmed active (CA). Compounds not meeting these criteria were listed as confirmed inactive (CI). We used a total of 37,169 total compounds, of which 325 belong to class CA, 875 are of class CM, and the remaining 35,969 are of class CI. In order to carry out tests on different sizes of the focus data set we combined these compounds as follows: In all our tests, we adopted a threshold ($thres = 0.5$) such that the CA class corresponds to the focus data set and classes CM and CI constitute the complement data set.

Experimental tests have been carried out on a network of Linux workstations[2] located in different LANs of the University of Konstanz. The software has been developed in Java and 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 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.

4.2 Running Time and Speedup

We carried out the analysis of the distributed approach with the RRP policy by showing the speedup curve of the parallel over the serial algorithm for different minimum support values ($minSupp$).

The performance of the sequential algorithm that is used to determine the speedup, is obtained from an optimized version of the algorithm described in [8] in the highest performing server[3] that is available for our tests.

As shown in Fig. 6a, the speedup is linear in the first part of the chart, especially for the lowest $minSupp$ value (4 percent). For $minSupp$ values 6 percent and 8 percent, 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. Moreover, it means that the determination of the optimal number of processors for a given task is not critical with respect to the running time.

As expected, the algorithm provides better performance for lower values of the minimum support, i.e., for larger problem sizes, where the speedup monotonically increases till the maximum number of the available computing nodes. The figure also shows the running time for $minSupp = 4$ percent for completeness.

In Fig. 6b, we provide a comparison of the speedup curves of the different DLB methods. In the first part of the chart, all methods are equivalent. This typically corresponds to a high parallel efficiency and a high ratio of the problem size over the number of processors. In the second part, however, it is evident that, for the given problem size, the RRP method is able to exploit more computational resources; the speedup always increases when the number of processors increases. In contrast, the MS method shows evident scalability issues; from 48 to 64 processors there is a drop of the speedup, hence the overall time increases. The two RP-based methods seem to be able to keep the speedup at least constant, but they are not able to effectively exploit more resources. It should be noted that the RP-based methods actually had a high variability in the performance; the reported values are averages over several trials. Anyway, in a single trial, they have never performed better than the RRP method and their speedup typically oscillated between this and the MS method.

4.3 Dynamic Load Balancing Evaluation

The speedup, or equivalently the running time, gives an immediate measure of the effectiveness of the different DLB methods; a comparison of the speedup curves has clearly

2. 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.

3. AMD Opteron dual Processor 848 2193MHz with 1MB cache and 32GB RAM.

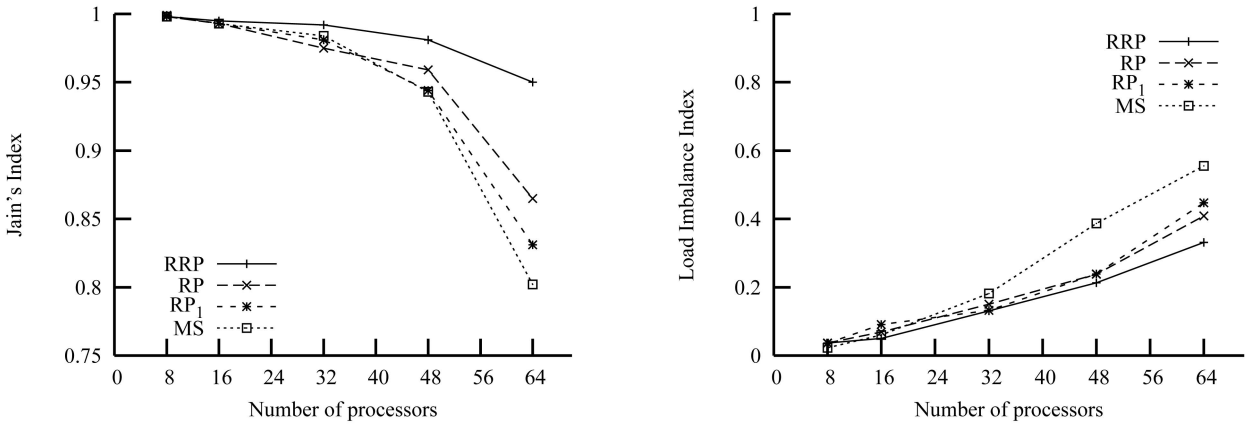

Fig. 7. Overall DLB evaluation ($minSupp = 4$ percent). (a) Jain's index (higher values are better) and (b) load imbalance index (lower values are better).

shown (Fig. 6b) that the distributed application based on the RRP method has a better performance than the others.

In this section, we have a closer look at the reasons in order to better understand the differences among the DLB approaches.

We first compare two overall DLB performance figures, namely, the Jain's fairness index [28] and the relative load imbalance index.

The Jain's fairness index is often adopted to measure the equality of the allocation of a shared resource to different contending entities. In this case, we want to measure the quality of the load balancing, which aims at the equal allocation of the overall computational load to processors. The Jain's fairness index is defined as:

$$J = f(x_1, x_2, \ldots, x_N) = \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N \cdot \sum_{i=1}^{N} x_i^2}, \quad (1)$$

where x_i is a measure of the work load at processor P_i. In particular, the quantity we consider is the running time spent in useful work by each processor:

$$x_i = T_{P_i,work}.$$

The index of (1) is continuous and bounded in the range $[0, 1]$. It is independent of the scale, the metric and the total amount of the shared resource and of the number of contending entities. An index closer to 1 means a better fairness in the allocation. In our context, the index is a measure of the quality of the load balance.

Another index that has been adopted to measure the DLB performance (e.g., in [29]) is the relative load imbalance index (LI), given by

$$LI = f(x_1, x_2, \ldots, x_N) = 1 - \frac{\sum_{i=1}^{N} x_i}{N \cdot \max_{i=1}^{N} x_i}. \quad (2)$$

The index is bounded in the range $[0, 1]$, but it does not have the other properties described above for the Jain's index. The LI index is a measure of the load imbalance; a lower value means better load balancing.

The charts in Fig. 7 show these two performance figures for the different DLB algorithms. The RRP method confirms its superiority, especially when the number of processing elements increases and the system has low parallel efficiency (ratio between the speedup and the number of processors). The MS method performs worse than the others and the two RP-based methods have a similar intermediate behavior.

In general, the charts in Fig. 7 do not add much information to the one already provided by the speedup comparison. However, the Jain's index shows a small superiority of the RP-based method with the heuristic approach (RP) over the simple work splitting technique (RP_1).

Finally, we look at the relative contribution of the two main components of the DLB strategy, namely the donor selection and the work splitting mechanism. To evaluate the quality of the donor selection we consider the idling periods of the workers, since they can be related to a wrong choice of the donor. For the evaluation of the work splitting mechanism, we compute the relative contribution of the computation overhead to the working time in the overall distributed application. The two efficiency indices, respectively, for the donor selection (E_{DS}) and the work splitting (E_{WS}), are defined as follows:

$$E_{DS} = \sum_{i}^{N} \frac{T_{P_i,idle}}{T_{P_i,work} + T_{P_i,idle}} \quad (3)$$

and

$$E_{WS} = \sum_{i}^{N} \frac{T_{P_i,comp.overhead}}{T_{P_i,work}}. \quad (4)$$

Fig. 8a shows the donor selection efficiency for different numbers of processing nodes. The RRP policy manages to keep the workers more busy than the other policies, the two RP-based approaches behave very similarly and the MS method clearly shows its scalability limitations. This is another confirmation of the results analyzed so far. However, in the chart of Fig. 8b, the MS method is able to

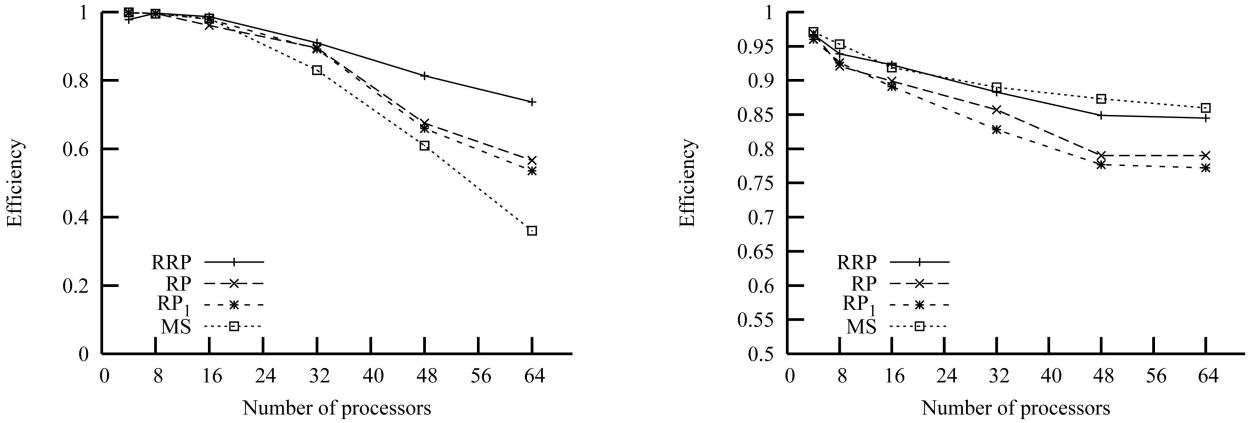

Fig. 8. Efficiency evaluation of DLB mechanisms ($minSupp = 4$ percent). (a) Donor selection efficiency E_{DS} (higher values are better) and (b) work splitting efficiency E_{WS} (higher values are better).

outperform all the others. This is an evident effect of the careful selection of the donor in the centralized approach with global knowledge. Job requests issued by the master are addressed to a proper donor with a small probability to generate trivial tasks. The RRP policy, however, is able to reach a good efficiency, close to the optimal of the MS method.

This final analysis confirms that the proposed ranked-random polling DLB policy is able to combine good scalability properties and low computation overheads by exploiting global knowledge of job statistics.

5 Conclusions

In this paper, we presented a peer-to-peer computing approach to the frequent subgraph mining problem. The distributed algorithm has been applied to the task of discriminative molecular fragment discovery. Several issues have been discussed for an effective design of a large-scale distributed approach to the frequent subgraph mining problem. The adopted approach is based on three components, which are a partitioning criterion of the search space, a novel dynamic load balancing policy and a peer-to-peer communication architecture. Very low communication and synchronization requirements, a decentralized receiver-initiated load balancing and high scalability of the communication framework make this distributed data mining application suitable for large-scale, multidomain, nondedicated heterogeneous environments like computational grids. Furthermore, the proposed approach naturally tolerates node failures and communication latency and supports dynamic resource aggregation. Experimental tests on real molecular compounds in a distributed nondedicated computing environment confirmed its effectiveness in terms of running time performance, low parallel overhead and load balancing.

Future research effort will focus on very large-scale systems, where the centralized server for job statistics could potentially become a bottleneck. In this case, the dynamic load balancing framework needs to be improved with a distributed management of job statistics. For example, the DLB framework could be extended in a hierarchical structure resembling the Domain Name Service infrastructure or a second-generation peer-to-peer system. Moreover, in the particular application we presented, the focus data set is relatively small and can be easily duplicated, while the complement data set is typically large. A data-parallel approach based on the partitioning of the complement data set could provide significant improvements and allow the mining of very large data sets that do not fit in the main memory of a single system. This data-parallel approach can be integrated into the proposed task-parallel one in order to fully exploit a distributed memory system.

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." The authors 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," *Proc. IEEE Int'l Conf. Data Mining (ICDM '03)*, Nov. 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," *Proc. 1993 ACM SIGMOD Int'l Conf. Management of Data*, May 1993.

[4] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New Algorithms for Fast Discovery of Association Rules," *Proc. Third Int'l Conf. Knowledge Discovery and Data Mining (KDD '97)*, pp. 283-296, 1997.

[5] R. Finkel and U. Manber, "DIB—A Distributed Implementation of Backtracking," *ACM Trans. Programming Languages and Systems*, vol. 9, no. 2, pp. 235-256, Apr. 1987.

[6] M. Deshpande, M. Kuramochi, and G. Karypis, "Automated Approaches for Classifying Structures," *Proc. Workshop Data Mining in Bioinformatics (BioKDD)*, pp. 11-18, 2002.

[7] X. Yan and J. Han, "gSpan: Graph-Based Substructure Pattern Mining," *Proc. IEEE Int'l Conf. Data Mining (ICDM '02)*, 2002.

[8] C. Borgelt and M.R. Berthold, "Mining Molecular Fragments: Finding Relevant Substructures of Molecules," *Proc. IEEE Int'l Conf. Data Mining (ICDM '02),* pp. 51-58, Dec. 2002.
[9] S. Kramer, L. de Raedt, and C. Helma, "Molecular Feature Mining in HIV Data," *Proc. Seventh Int'l Conf. Knowledge Discovery and Data Mining, (KDD '01),* pp. 136-143, 2001.
[10] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.
[11] E.M. Luks, "Isomorphism of Graphs of Bounded Valence Can Be Tested in Polynomial Time," *J. 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," *Proc. IEEE DM-Grid Workshop Int'l Conf. Data Mining (ICDM '04),* Nov. 2004.
[14] C. Wang and S. Parthasarathy, "Parallel Algorithms for Mining Frequent Structural Motifs in Scientific Data," *Proc. 18th Ann. Int'l Conf. Supercomputing (ICS '04),* June 2004.
[15] F. Schreiber and H. Schwöbbermeyer, "Towards Motif Detection in Networks: Frequency Concepts and Flexible Search," *Proc. Int'l Workshop Network Tools and Applications in Biology (NETTAB '04),* pp. 91-102, Sept. 2004.
[16] G. Di Fatta and M.R. Berthold, "High Performance Subgraph Mining in Molecular Compounds," *Springer's LNCS Proc. 2005 Int'l Conf. High Performance Computing and Comm. (HPCC-05),* Sept. 2005.
[17] G. Di Fatta and M.R. Berthold, "Efficient Mining of Discriminative Molecular Fragments," *Proc. 17th IASTED Int'l Conf. Parallel and Distributed Computing and Systems (PDCS-05),* Nov. 2005.
[18] R. Agrawal and J. Shafer, "Parallel Mining of Association Rules," *IEEE Trans. Knowledge and Data Eng.,* vol. 8, no. 6, pp. 962-969, Dec. 1996.
[19] E. Han, G. Karypis, and V. Kumar, "Scalable Parallel Data Mining for Association Rules," *IEEE Trans. Knowledge and Data Eng.,* vol. 12, no. 3, pp. 337-352, May/June 2000.
[20] Daylight Chemical Information Systems, Inc., SMILES—Simplified Molecular Input Line Entry Specification, http://www.daylight.com/smiles, 2006.
[21] R. Karp and Y. Zhang, "A Randomized Parallel Branch-and-Bound Procedure," *Proc. 20th Ann. ACM Symp. Theory of Computing (STOC '88),* pp. 290-300, 1988.
[22] S. Chakrabarti, A. Ranade, and K. Yelick, "Randomized Load-Balancing for Tree-Structured Computation," *Proc. Scalable High Performance Computing Conf. (SHPCC '94),* pp. 666-673, May 1994.
[23] Y. Chung, J. Park, and S. Yoon, "An Asynchronous Algorithm for Balancing Unpredictable Workload on Distributed-Memory Machines," *ETRI J.,* vol. 20, no. 4, pp. 346-360, Dec. 1998.
[24] V. Kumar, A. Grama, and V.N. Rao, "Scalable Load Balancing Techniques for Parallel Computer," *J. Parallel and Distributed Computing,* vol. 22, no. 1, pp. 60-79, July 1994.
[25] A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to Parallel Computing.* Addison-Wesley, 2003.
[26] Nat'l Cancer Inst., DTP AIDS Antiviral Screen Dataset, http://dtp.nci.nih.gov/docs/aids/aids/data.html, May 2004.
[27] 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," *J. Nat'l Cancer Inst.,* vol. 81, pp. 577-586, 1989.
[28] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems," DEC Research Report TR-301, Technical Report, 1984.
[29] R. Sakellariou and J.R. Gurd, "Compile-Time Minimisation of Load Imbalance in Loop Nests," *Proc. 11th ACM Int'l Conf. Supercomputing (ICS '97),* pp. 277-284, July 1997.

Giuseppe Di Fatta received the masters degree in electronic engineering in 1995 and the PhD degree in electronic, computer science, and telecommunications engineering in 2002 from the University of Palermo (Italy). In 1999, he was with the International Computer Science Institute, Berkeley, California, as research fellow. Since 2000, he has joined the High Performance Computing and Networking Institute of the Italian National Research Council (ICAR-CNR) and he has been a lecturer at the University of Palermo (Italy). Since March 2004, he has been with the University of Konstanz, Germany. His research interests include distributed computing, computer networks, data mining, and soft computing. He is a member of the IEEE.

Michael R. Berthold spent more than seven years in the US, among others, at Carnegie Mellon University, Intel Corporation, the University of California at Berkeley, and—most recently—as director of an industrial think tank in South San Francisco. Since August 2003, he has been at the ALTANA-Chair for Bioinformatics and Information Mining at Konstanz University, Germany, where his research focuses on using machine learning methods for the interactive analysis of large information repositories in the Life Sciences. He is past president of the North American Fuzzy Information Processing Society, associate editor of several journals, and on the Board of Governors of the IEEE System, Man, and Cybernetics Society. He has been involved in the organization of various conferences, most notably the IDA-series of symposia on Intelligent Data Analysis. Together with David Hand, he coedited the successful textbook *Intelligent Data Analysis: An Introduction*, which has recently appeared in a completely revised, second edition. He is a senior member of the IEEE.