Improving Scalability in a Scientific Discovery System by Exploiting Parallelism

Gehad Galal, Diane J. Cook and Lawrence B. Holder
University of Texas at Arlington
{galal,cook,holder}@cse.uta.edu

Abstract

The large amount of data collected today is quickly overwhelming researchers' abilities to interpret the data and discover interesting patterns. Knowledge discovery and data mining approaches hold the potential to automate the interpretation process, but these approaches frequently utilize computationally expensive algorithms.

This research outlines a general approach for scaling KDD systems using parallel and distributed resources and applies the suggested strategies to the SUBDUE knowledge discovery system. SUBDUE has been used to discover interesting and repetitive concepts in graph-based databases from a variety of domains, but requires a substantial amount of processing time. Experiments that demonstrate scalability of parallel versions of the SUBDUE system are performed using CAD circuit databases and artificially-generated databases, and potential achievements and obstacles are discussed.

Introduction

One of the barriers to the integration of scientific discovery methods into practical data mining approaches is their lack of scalability. Many scientific discovery systems are motivated from the desire to evaluate the correctness of a discovery method without regard to the method's scalability. As an example, our SUBDUE system was developed to evaluate the effectiveness of the minimum description length principle to discover regularities in a variety of scientific domains (Cook, Holder, & Djoko 1996).

Another factor is that some scientific discovery systems deal with richer data representations that only degrade scalability. A number of attribute-value-based approaches have been developed that discover concepts and can address issues of data relevance, missing data, noise, and utilization of domain knowledge. However, much of the data being collected is structural in nature, requiring tools for the analysis and discovery of concepts in structural data (Fayyad *et al.* 1996). For example, the SUBDUE system uses a graph-based representation of the input data that captures the structural information. Although the subgraph isomorphism procedure needed to deal with this data has been polynomially constrained within SUBDUE, the system still spends a considerable amount of computation performing this task.

The goal of this research is to demonstrate that KDD systems can be made scalable through efficient use of parallel and distributed hardware. To accomplish this goal, we introduce a general approach to parallelizing KDD systems and apply the proposed techniques to the SUBDUE discovery system.

Related approaches to scaling data mining and discovery systems have been pursued. Parallel MIMD approaches to concept learning have included partitioning the data set among processors and partitioning the search space among available processors (Provost & Hennessy 1996; Chan & Stolfo 1993). Data partitioning approaches have also been effective for certain limited approaches to data mining and knowledge discovery on SIMD architectures. Improving the scalability of scientific discovery systems will help break down the barrier excluding these techniques from practical data mining approaches.

Overview of SUBDUE

We have developed a method for discovering substructures in databases using the minimum description length principle. SUBDUE discovers substructures that compress the original data and represent structural concepts in the data. Once a substructure is discovered, the substructure is used to simplify the data by replacing instances of the substructure with a pointer to the newly discovered substructure.

The substructure discovery system represents structural data as a labeled graph. Objects in the data

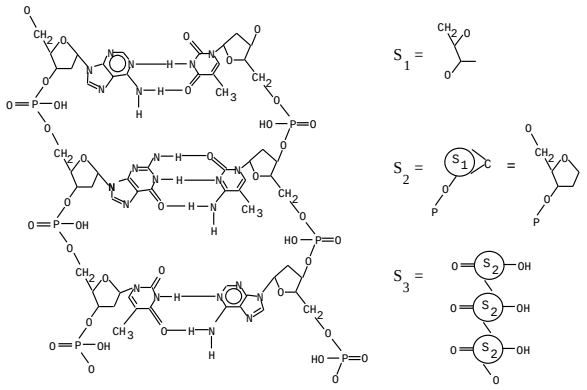

Figure 1: Sample results of Subdue on a protein sequence.

map to vertices or small subgraphs in the graph, and relationships between objects map to directed or undirected edges in the graph. A *substructure* is a connected subgraph within the graphical representation. This graphical representation serves as input to the substructure discovery system. An *instance* of a substructure in an input graph is a set of vertices and edges from the input graph that match the graphical representation of the substructure.

Figure 1 shows a sample input database containing a portion of a DNA sequence. In this case, atoms and small molecules in the sequence are represented with labeled vertices in the graph, and the single and double bonds between atoms are represented with labeled edges in the graph. SUBDUE discovers substructure S_1 from the input database. After compressing the original database using S_1, SUBDUE finds substructure S_2, which when used to compress the database further allows SUBDUE to find substructure S_3. Such repeated application of SUBDUE generates a hierarchical description of the structures in the database.

The substructure discovery algorithm used by SUBDUE is a beam search. The algorithm begins with the substructure matching a single vertex in the graph. Each iteration the algorithm selects the best substructure and incrementally expands the instances of the substructure. The algorithm searches for the best substructure until all possible substructures have been considered or the total amount of computation exceeds a given limit. Evaluation of each substructure is determined by how well the substructure compresses the description length of the database.

Because instances of a substructure can appear in different forms throughout the database, an inexact graph match is used to identify substructure instances with a bounded amount of variation from the substructure definition. A scientist can direct the search with background knowledge in the form of known substructure models that may potentially appear in the database, or graph match rules to adjust the cost of each inexact graph match test. SUBDUE has been successfully applied to databases in domains including image analysis, CAD circuit analysis, Chinese character databases, program source code, chemical reaction chains, Brookhaven protein databases, and artificially-generated databases (Cook, Holder, & Djoko 1996).

The results of the scalability study in this paper are demonstrated on databases in two different domains. The first type of database is a graph representation of CAD A-to-D converer circuit provided by National Semiconductor containing 8,441 edges and 19,206 edges. The second type of database is an artificially-constructed graph with 1,000 vertices and 2,500 edges in which instances of a predefined substructure are embedded in a random graph. To test scalability on larger databases while maintaining the characteristics of these two domains, we generate multiple copies of the CAD and ART graphs and merge the copies together by arbitrarily connecting the individual graphs. The terms "n CAD" and "n ART" thus refer to a graphs consisting of n merged copies of the CAD or ART graphs.

Scaling KDD Systems

Making use of parallel and distributed resources can significantly affect the scalability of a KDD system. Parallelizing a knowledge discovery system is not easy because many KDD systems rely upon heuristics and greedy algorithms to avoid the intractability inherent in an exhaustive approach. Both heuristics and greedy algorithms share the potential of finding a suboptimal solution and, on closer inspection, a sequentially oriented solution. In many cases KDD algorithms can perform better if they are provided with enough history of the problem being solved, thus they will perform better in a sequential approach. In addition, the knowledge discovered in each step by KDD systems depends heavily on what has been discovered in previous steps. Thus, we cannot decompose the work without increasing the synchronization and communication between the parallel processors.

Two main MIMD distributed memory approaches to designing parallel algorithms are the functional parallel approach and the data parallel approach. In the functional parallel approach the algorithm steps are assigned to different processors, while in a data parallel approach each processor applies the same algorithm to different parts of the input data.

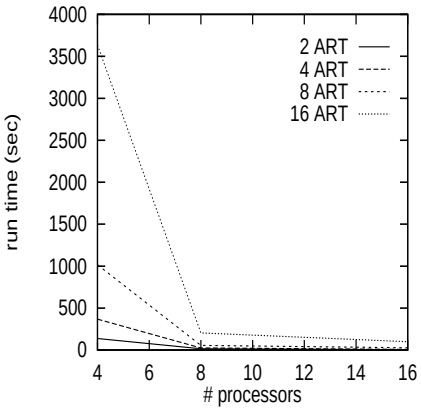

Figure 2: Discovery time of 60 substructures in ART.

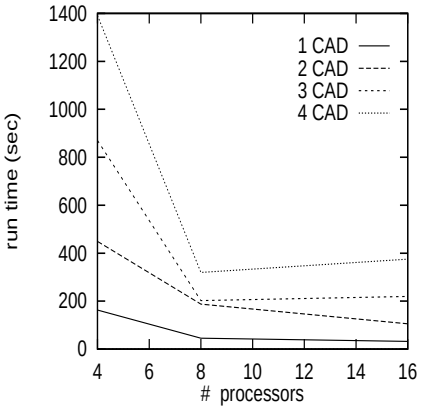

Figure 3: Discovery time of 60 substructures in CAD.

Functional Parallel SUBDUE

The main idea behind this algorithm is to divide SUBDUE's search for candidate substructures among processors. The search queue is maintained by one master processor which keeps track of the best discovered substructures. The master decides whether to keep expanded substructures based on a global evaluation of discovered substructures. If a slave processor does not have any substructures then the master asks another processor to transfer a substructure to the requesting processor.

Figures 2 through 3 graph the decrease in runtime of FP-SUBDUE as the number of processors increases using an nCUBE 2. The amount of compression achieved may also sometimes increase as the number of processors increases. This is due to the fact that the beam width combined over all processors is greater than a single beam width on the serial machine, and thus a greater number of substructures can be considered.

Dynamic Partitioning SUBDUE

In the first data partitioning approach, Dynamic-Partitioning SUBDUE, each processor starts evaluating a disjoint set of the input data. When DP-SUBDUE is run, processor i begins processing a candidate substructure corresponding to the ith unique label in the graph. Each processor receives a copy of the entire input graph and processes a portion of the possible substructures. To prevent work replication, DP-SUBDUE constrains processors expanding a substructure to only include vertices with a label index greater than the processor ID. Load balancing is permitted between processors to prevent excessiving processor idling. The partitions here are logical: the set of all instances of all the candidate substructures discovered by a processor constitutes its partition. Quality control is imposed on the processors in the DP-SUBDUE system by periodically pruning all substructure candidates with values less than the global average.

Results from the DP-SUBDUE system indicate that very limited speedup can be achieved by distributing the substructure expansion and evaluation. The work done to limit duplicate work and to load balance the system consumes considerable time in processing and communication. In addition, the memory requirements of this data partitioning approach are excessive because the entire database is copied on each processor. The speedup achieved is very limited and the results are not included in this paper.

Static Partitioning SUBDUE

Although the DP-SUBDUE approach was not successful, the data partitioning idea itself is very appealing in terms of both memory usage and speedup. Here we introduce a static partitioning parallel approach.

In SP-SUBDUE we partition the input graph into n partitions for n processors. Each processor performs sequential SUBDUE on its local graph partition and broadcasts its best substructures to the other processors. Each processor then evaluates the communicated substructures on its own local partition. Once all evaluations are complete, a master processor gathers the results and determines the global best discoveries.

In partitioning the graph we want to balance the work load equally between processors while retaining as much information as possible (edges along which the graph is partitioned may represent important information). The *Metis* graph partitioning package tries to partition the graph so that the sum of the cut edges is minimized. The run time of Metis to partition the databases is very small (ten seconds on average) and is thus not included in the parallel run time.

Figures 4 and 5 graph the run time of SP-SUBDUE

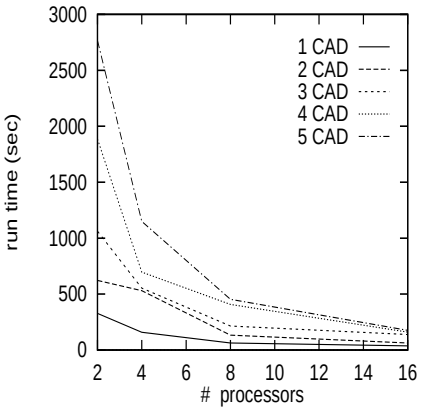

Figure 4: CAD database evaluation time.

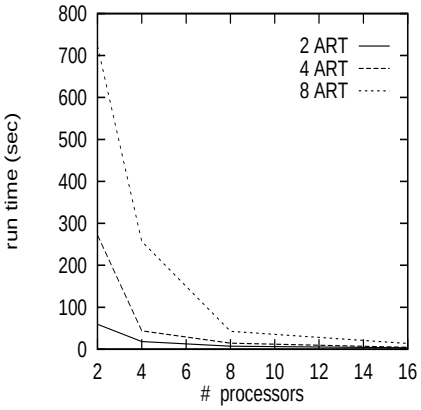

Figure 5: ART database evaluation time.

on the CAD and artificial databases as the number of processors increases. The speedup achieved with the ART database is always superlinear. This is because the run time of sequential SUBDUE is nonlinear with respect to the size of the database. Each processor essentially executes a serial version of SUBDUE on a small portion of the overall database, so the combined run time is less than that of serial SUBDUE.

Increasing the number of processors for the nCAD and nART databases results in similar quality discovered substructures. As the number of partitions becomes large, the quality of the discovered substructures will decrease because some of the edges are cut. However, with a small number of partitions superior compression to that of the sequential version can be realized because the combined beam length is larger over several processors than for the one processor used in the serial version of the algorithm.

Because the data is partitioned among the processors, SP-SUBDUE can also utilize the increased memory resources of a network of workstations using communication software such as PVM. The performance of SP-SUBDUE on a network of 14 PCs also improves close to linearly in the number of processors.

Conclusions

The increasing structural component of today's databases requires data mining algorithms capable of handling structural information. The SUBDUE system is specifically designed to discover knowledge in structural databases. However, the computational expense of a discovery system such as SUBDUE can deter widespread application of the algorithms.

In this paper, we investigate methods for improving the scalability of scientific discovery methods using parallel resources. When comparing the benefits of the three parallel applied to SUBDUE, approaches, DP-SUBDUE is discarded because of poor run time and heavy memory requirements. FP-SUBDUE can prove effective in discovering substructures in very large databases due to its unique search algorithm. SP-SUBDUE is the most interesting approach of all – by partitioning the database effectively, SP-SUBDUE proves to be a highly scalable system. One of our databases contains 2 million vertices and 5 million edges, yet SP-SUBDUE is able to process the database in less than three hours. The minimal amount of communication and synchronization that is required make SP-SUBDUE ideal for distributed environments. We have demonstrated the scalability of one KDD system using these techniques, and will continue to apply the described methodology to other systems.

Acknowledgements

This research is supported by NASA grant NAS5-32337 and NSF grant IRI-9502260.

References

Chan, P., and Stolfo, S. 1993. Toward parallel and distributed learning by meta-learning. In *Working notes of the AAAI-93 workshop on Knowledge Discovery in Databases*, 227–240.

Cook, D. J.; Holder, L. B.; and Djoko, S. 1996. Scalable discovery of informative structural concepts using domain knowledge. *IEEE Expert* 11(5):59–68.

Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurusamy, R. 1996. *Advances in Knowledge Discovery and Data Mining.* Menlo Park, CA: AAAI Press.

Provost, F. J., and Hennessy, D. 1996. Scaling up: Distributed machine learning with cooperation. In *Proceedings of AAAI*, 74–79.