High-Performance Combinatorial Algorithms

Ali Pınar

Computational Research Division
Lawrence Berkeley National Laboratory
Email: apinar@lbl.gov

Introduction

Combinatorial algorithms have long played an important role in many applications of scientific computing such as sparse matrix computations and parallel computing. The growing importance of combinatorial algorithms in emerging applications like computational biology and scientific data mining calls for development of a high performance library for combinatorial algorithms. Building such a library requires a new structure for combinatorial algorithms research that enables fast implementation of new algorithms. We propose a structure for combinatorial algorithms research that mimics the research structure of numerical algorithms. Numerical algorithms research is nicely complemented with high performance libraries, and this can be attributed to the fact that there are only a small number of fundamental problems that underlie numerical solvers. Furthermore there are only a handful of kernels that enable implementation of algorithms for these fundamental problems. Building a similar structure for combinatorial algorithms will enable efficient implementations for existing algorithms and fast implementation of new algorithms. Our results will promote utilization of combinatorial techniques and will impact research in many scientific computing applications, some of which we list below.

Sparse Linear and Non-linear Solvers

Solving sparse linear and nonlinear equations is at the heart of many DOE applications, such as accelerator modeling, astrophysics, nanoscience, and combustion, among others. Direct methods for solving sparse linear equations are widely used in many applications such as inversion operator for the shift-and-invert algorithms for high accuracy eigencomputations, solution of coarse grid problems as part of a multigrid solver, and subdomain solutions in domain decomposition methods. The performance of direct methods relies heavily on the ordering of rows and columns of the matrix. Convergence of iterative methods on the other hand, is strongly affected by the preconditioners, which is another field that requires combinatorial tools. Incomplete factorizations are frequently used for preconditioning, and their effectiveness can be enhanced by cleverly ordering the matrix to minimize the effect of the discarded nonzeros. By permuting large entries of the matrix to the main diagonal, the convergence of iterative solvers can be accelerated. Having large entries on the diagonal also can help to avoid the overhead of pivoting during factorizations. A final example is the novel support theory for preconditioning, which translates the preconditioning problem into a graph theory problem.

Computational Biology

Another area where combinatorial algorithms arise frequently is computational biology. DNA, RNA, and proteins can be modeled as sequences over small alphabets. The DNA shotgun sequencing method is based on constructing the whole DNA from its fragments, and relies on dynamic programming algorithms to detect if and how the DNA fragments overlap. Another important problem is the sequence alignment problem, which identifies similar subsequences between two sequences. Multiple sequence alignment does the same

thing with more than two sequences, and remains a combinatorially challenging problem. Sequence alignment results are used to construct phylogenetic trees, which is another problem where combinatorial methods are employed. Contact maps are used to represent the three-dimensional structure of proteins, by including information of particles within a small distance, in addition to the sequence information. Contact map alignment is used to measure similarity between the three-dimensional structures, and is another challenging combinatorial problem. Molecules can be considered as graphs where atoms correspond to vertices and chemical bonds correspond to edges. A search for specific structures can then be studied as a graph isomorphism problem. Understanding gene regulatory networks is of central importance to DOE's biology mission. Gene regulatory networks and other biological networks such as protein interaction networks, metabolic networks, and signaling networks can be modeled as graphs, where vertices represent biological entities and processes (such as genes, proteins, and chemical reactions) and edges represent interactions between entities or input/output relationships. Since paths on these graphs explain biological processes, analysis of these paths can help us understand and control how these networks work. Subgraph isomorphism algorithms are also used on these networks for chemical information retrieval. As can be seen from the examples above, computational biology is incredibly rich in terms of the number and variety of combinatorial problems it includes.

Electric Power Systems

Power system analysis is another field that will benefit from novel combinatorial algorithms. The *unit commitment problem*, a traditional challenge for power systems, is the task of determining an optimal set of generators to commit to operation over a set time-period, typically a day or week in advance of use. The solution should take costs, plant limitations, network topology and capacity constraints, expected load requirements, and security constraints into account. This problem is fundamentally combinatorial in nature. Another interesting problem is the *optimal power flow problem* that determines how the power should flow from the generators. These models should include devices with discrete settings, thus the solution methods require combinatorial analysis. Arguably the most important problem in power system analysis is dynamic security assessment, where we want to determine whether a system, represented by a dynamical model, will survive a set of possible contingent outages, or if it will fail in some catastrophic manner. As evidenced by the recent power crisis in the Northeast, this problem is critically important, and it can be computationally extensive when all possible contingencies are considered, which expands the search space exponentially. Algorithms that cleverly search this space will improve efficiency and will clearly benefit from combinatorial techniques.

Scientific Data Mining

Advances in technology have enabled the production of massive volumes of data through observations and simulations in many applications such as biology, high-energy physics, and astrophysics. These new data sets and the associated queries are significantly different than those of the traditional database systems, and pose new challenges for efficient storage and retrieval of data, which requires novel combinatorial techniques. Moreover, automated tools for discovering information from these massive data sets are critically important. These data mining tools commonly adopt geometric or graph theoretical models. Information retrieval from graph-based sources requires many new graph algorithms. A query, for instance, may be expressed in the form of a subgraph, called a template, and all instances of this subgraph in the database may need to be returned. Alternatively, a query may be in the form of two vertices, and the minimal subgraph, according to some metric, that connects these two vertices needs to be returned. In general, these graph-based sources have very small graph diameters. In addition, the data sets are often very large, requiring parallel computation.

Utilizing the Computational Infrastructure

Another important application of combinatorial algorithms is the efficient utilization of the underlying computational infrastructure. Even for applications where problems are modeled with techniques of continuous mathematics, we have to decompose the problem into subproblems and map them onto processors, and schedule the tasks to satisfy precedence constraints in a parallel computer, which are all combinatorial problems. The increasing gap between CPU and memory performances argues for the design of new algorithms and data structures and data reorganization to improve locality at memory, cache, and register levels. The next generation petaflops architectures are expected to have orders of magnitude more processors. An increased number of processors, along with the increasing gap between processor and network speeds, will expose some of the limitations of the existing approaches. Novel decomposition techniques and interprocessor communication algorithms will be required to cope with evolving architectures.

Proposed Structure for Combinatorial Algorithms Research

The examples of the preceding section can be extended, or discussed in more detail, yet two points already stand out. First, discrete mathematics and combinatorial algorithms will play an increasingly important role in many DOE applications, with a leading role in emerging applications such as computational biology and scientific data mining, as opposed to its supportive role in the traditional applications of scientific computing. Secondly, the breadth of expertise required to work on these problems is very wide. Building a high performance solver for a biology problem requires expertise in biology, discrete mathematics, and high performance computing. The numerical algorithms community faced a similar situation, and their success motivates us to further look at how they are organized. Numerical algorithms interact with the applications through some well-defined models like differential equations, optimization, or linear algebra problems. Thus, it is important to identify model combinatorial problems, where combinatorial algorithms and the applications will interface. It is also worth noting that fundamental numerical algorithms research is nicely complemented with solid high-performance libraries. What makes this possible is that many numerical algorithms rely on a small number of fundamental linear algebra problems, such as solving systems of linear equations, eigensolvers, and singular-value decompositions. Furthermore, solutions to these fundamental linear algebra problems are built on only a handful of kernel operations such as matrix-matrix, matrix-vector, and vector-vector multiplications. Implementations of these few kernels can then be used to implement a wide variety of other algorithms. High performance is achieved via efficient implementations of such kernels, and existing algorithms can often be tuned to cope with architectural advances by only updating these kernels. Moreover, new algorithms built on these kernels can be easily implemented efficiently, enabling researchers to focus only on numerical and mathematical aspects of the algorithms.

Given the success of this approach by the numerical algorithms community, we propose a similar roadmap. For this purpose, we need to identify model problems that will serve as the interface between combinatorial algorithms and the applications, akin to differential equations for numerical algorithms. These problems must provide the necessary modeling power for the applications, thus their identification requires close interactions with application scientists. Next, we need to identify fundamental combinatorial algorithms that might serve as building blocks for more sophisticated algorithms. Finally, we need to identify the kernel operations that will enable efficient implementations of the fundamental algorithms on high performance computing platforms. Identifying the basic solution methods and associated kernels will be a community effort for the combinatorial scientific computing community.

Our research will lead to efficient solvers for combinatorial algorithms on high performance computing platforms, which will increase the utilization of combinatorial techniques in many DOE scientific computing applications. More importantly, the structure we are proposing for combinatorial algorithms research will impact the future of combinatorial scientific computing. This structure will promote *high quality*, since algorithms will be implemented with well-tuned kernels, and *high productivity*, since such kernels will provide rapid implementations of new algorithms and enable researchers to concentrate only on combinatorial aspects of the problems. The use of standard models is more mature in some areas like parallel scientific computing and sparse matrix computations, whereas models are slowly developing for other fields like computational biology and scientific data mining. For instance, graph partitioning is a widely accepted model for load balancing, because it provides a useful model for the applications to represent their computational loads and communication requirements.

Our efforts in this project can be grouped under two areas: The first is our collaboration with application scientists to identify combinatorial models and associated problems that will influence emerging applications, and the second is identifying kernels for algorithms that are frequently used with the standard models. These kernels must not only serve as building blocks for a large number of algorithms, but also achieve efficient implementations for high-performance platforms. We want to stress that our efforts will not only be decomposing algorithms into kernels and implementation of the kernels, but will also involve designing new algorithms for those problems whose kernels are not suitable for high performance platforms. For instance, Tarjan's linear-time algorithm to find strongly connected components of a graph is based on depth-first search, which is provably hard to parallelize. We designed a novel divide-and-conquer algorithm, which is asymptotically slower, but amenable to parallelization since it uses reachability as the kernel operation. It will be necessary to redesign some of the algorithms to achieve scalability on high performance computing platforms, even at the cost of increased asymptotic complexity.

A Sample Problem

We will use the maximum flow problem as an example to demonstrate how a solution to a combinatorial problem can affect a wide variety of applications. In a maximum-flow problem, we are given a graph with edge capacities, a source and a sink, and we wish to find a maximum flow that satisfies the edge capacities from the source to the sink. By duality, a maximum-flow solution defines a minimum cut, i.e., a set of edges with minimum cardinality (or with minimum total capacity) that disconnects the graph. It is better to look at the maximum flow problem as a family of problems as opposed to a single problem, because there are many variations, such as the minimum-cost maximum-flow problem. In the minimum-cost, maximum-flow problem, costs are assigned to edges, and the problem is to push the flow from the source to the terminal in a cost-efficient way. One can envision even more variations with different definitions of the cost function. The algorithms for different instances of the maximum flow problem are similar in their combinatorial nature, and are mostly built on a small number of theoretical results.

Of interest to us are applications of the maximum-flow problem to scientific computing. Dynamic load balancing is already an important problem for parallel computing, and it will be even more important with the increasing number of processors for the next generation of supercomputers. The objective of dynamic load balancing is to achieve a balanced workload across the processors with minimal data movement. The diffusion method proposed by Cybenko is commonly used, but its shortcomings (high number of communicating neighbors, minimizing the 2-norm as opposed to the 1-norm) are exposed when it is used to generate multiple partitions for different phases of the computation. An alternative is a maximum flow formulation, where data reassignment corresponds to a flow from the overloaded processors to the

underloaded processors. The maximum flow model is a better formulation for the dynamic load balancing problem, since it enables handling other metrics that affect performance such as the number of communicating processor pairs.

Another application of the maximum-flow problem is exploiting flexibly assignable work (i.e., tasks that can be assigned to one of several processors without altering the volume of communication) to improve load balance. In many applications of parallel computing, distribution of data unambiguously implies distribution of work among the processors. But there are exceptions like molecular dynamics and overlapped domain decomposition applications, where some tasks can be assigned to one of several processors without altering the communication pattern. We showed that the problem of improving load balance with flexibly assignable tasks can be translated into a variant of the maximum flow problem.

Flow problems commonly arise in power systems analysis as well. The unit commitment problem mentioned above is a traditionally important problem for power systems. A subproblem of the unit commitment problem is the optimal power flow problem, which is a variant of the maximum flow problem. Minimum-cut techniques, on the other hand, can be used to detect weak spots of the power network, which would help us determine if a power system will survive a set of possible contingent outages. Although these problems are usually referred to as maximum flow problems, the dual problem of minimum cuts is equally important. For instance, divide-and-conquer is a fundamental algorithmic paradigm, and algorithms based on this paradigm are commonly used in many applications. Performance of many divide-and-conquer algorithms depends on the divide step of the algorithm, and requires decomposition into loosely coupled subproblems. Such decompositions can be achieved by graph partitioning tools, which employ minimum-cut algorithms to find minimum vertex-separators. One important application of this technique is the nested dissection orderings for sparse factorization of a matrix, where the matrix is decomposed into two independent submatrices by removal of a separator. The size of this separator directly effects how dense the factored matrix will be. We are currently investigating generalizations of this idea for ordering unsymmetric matrices. Recently, decomposition ideas have been applied to eigencomputations for very large matrices, and again performance of this novel approach depends on the sizes of separators used to decompose the graph. We have also used similar ideas to find a sparse null-space basis of a matrix. There are also applications of this idea on power systems for the optimum power flow problem.

The bipartite matching problem is a special case of the maximum-flow problem, and thus the algorithms for these problems are inherently similar. Weighted bipartite matching algorithms are used to permute large entries of a matrix to the main diagonal. One motivation for doing so is that having large entries on the diagonal accelerates convergence of iterative methods. Another motivation is that having large entries on the diagonal helps avoid the costly pivoting operations during sparse factorizations. Recently we have applied weighted bipartite matching algorithms to find a “nice” column basis for an overdetermined matrix. In general, bipartite matching algorithms are considered a “workhorse” in combinatorial optimization, and are used in the inner loop of more sophisticated solvers.

These examples show how important the maximum flow problem is and argue for the necessity of high performance maximum-flow solvers. However, there are two reasons it will not be enough to have one black box solver. First, practical performances of the algorithms depend on the input, and can often be enhanced by exploiting problem specific features. Secondly, we don't have a single problem, but variations of one basic problem. This is the reason why decomposing algorithms into kernels is important. When the kernels are available, it will be easy to implement variations of an algorithm for different instances of the problem.

For example, most maximum flow algorithms stem from the well-known maximum flow-minimum cut theorem. We proved a similar theorem for the new version of the maximum flow problem for the flexibly assignable tasks problem with the revised definitions for a cut and an augmenting path. This enables us to use any algorithm that is based on the original theorem, to solve the new version of the maximum-flow problem. And it is possible to replace specific kernels, when problem-specific information can be exploited for better efficiency.

We propose to conduct research in three areas related to the maximum flow problems. First, we will decompose the algorithms into kernels that will provide flexibility in terms of alternative implementations. Secondly, we will consider not only exact algorithms, but also approximation algorithms, since the latter might be required to limit preprocessing times for very large problems. And finally, we will investigate reductions to numerical problems, since numerical algorithms are generally more amenable to parallelization.

As we see, a maximum-flow solver will impact research in a wide variety of applications. Clearly, there will be even more applications that will introduce new variants of the maximum-flow problem, and it will be possible to solve these variants when the kernels are available. A similar case can be made for path analysis of graphs with its applications to sparse linear algebra, task scheduling for radiation transport equations, gene regularity networks, and scientific data mining. The path problems arising in these applications are not identical, but are similar in their combinatorial nature, and can be solved with similar kernel operations. It is possible to find a small set of combinatorial problems that can be used to solve problems in a wide variety of applications in need of combinatorial techniques. Thus it is possible to build a structure similar to that of numerical algorithms that will provide high quality tools and high productivity in various scientific computing applications and fundamental combinatorial algorithms research.

Conclusion

Combinatorial algorithms have long played an important role in scientific computing. This role is becoming even more critical with emerging applications like computational biology and scientific data mining. A high performance library of combinatorial algorithms will undoubtedly impact research activities for these applications. Building such a library requires research on new algorithms that are amenable for high performance platforms, but more importantly it requires a new structure for combinatorial algorithms research that enables fast implementation of new algorithms.

We propose adopting the numerical algorithms research structure with its proven success. We will identify model combinatorial problems and build solid solvers for these problems. Such solvers will encourage combinatorial modeling efforts, since models can quickly lead to solutions, and will increase utilization of combinatorial techniques. It is virtually impossible to build a solver for every single problem, since there are already many different problems, and new models will only generate more. However, these problems are not totally different, but can be grouped according to their combinatorial structure, and solvers for different versions of one problem can be built with similar kernel operations. Thus, we will provide not only solvers, but also kernels that enable solving new instances of the problem.

Our results will impact applications such as computational biology, sparse linear and nonlinear solvers, scientific data mining, and power engineering, via better utilization of combinatorial models and more flexible and more efficient solvers for combinatorial problems.