Preface

Graphs are among the most important abstract data structures in computer science, and the algorithms that operate on them are critical to modern life. Graphs have been shown to be powerful tools for modeling complex problems because of their simplicity and generality. For this reason, the field of graph algorithms has become one of the pillars of theoretical computer science, informing research in such diverse areas as combinatorial optimization, complexity theory, and topology. Graph algorithms have been adapted and implemented by the military and commercial industry, as well as by researchers in academia, and have become essential in controlling the power grid, telephone systems, and, of course, computer networks.

The increasing preponderance of computer and other networks in the past decades has been accompanied by an increase in the complexity of these networks and the demand for efficient and robust graph algorithms to govern them. To improve the computational performance of graph algorithms, researchers have proposed a shift to a parallel computing paradigm. Indeed, the use of parallel graph algorithms to analyze and facilitate the operations of computer and other networks is emerging as a new subdiscipline within the applied mathematics community.

The combination of these two relatively mature disciplines—graph algorithms and parallel computing—has been fruitful, but significant challenges still remain. In particular, the tasks of implementing parallel graph algorithms and achieving good parallel performance have proven especially difficult.

In this monograph, we address these challenges by exploiting the well-known duality between the canonical representation of graphs as abstract collections of vertices with edges and a sparse adjacency matrix representation. In so doing, we show how to leverage existing parallel matrix computation techniques as well as the large amount of software infrastructure that exists for these computations to implement efficient and scalable parallel graph algorithms. In addition, and perhaps more importantly, a linear algebraic approach allows the large pool of researchers trained in fields other than computer science, but who have a strong linear algebra background, to quickly understand and apply graph algorithms.

Our treatment of this subject is intended formally to complement the large body of literature that has already been written on graph algorithms. Nevertheless, the reader will find several benefits to the approaches described in this book.

(1) *Syntactic complexity.* Many graph algorithms are more compact and are easier to understand when presented in a sparse matrix linear algebraic format. An algorithmic description that assumes a sparse matrix representation of the graph, and operates on that matrix with linear algebraic operations, can be readily

understood without the use of additional data structures and can be translated into a program directly using any of a number of array-based programming environments (e.g., MATLAB®).

(2) *Ease of implementation.* Parallel graph algorithms are notoriously difficult to implement. By describing graph algorithms as procedures of linear algebraic operations on sparse (adjacency) matrices, all the existing software infrastructure for parallel computations on sparse matrices can be used to produce parallel and scalable programs for graph problems. Moreover, much of the emerging Partitioned Global Address Space (PGAS) libraries and languages can also be brought to bear on the parallel computation of graph algorithms.

(3) *Performance.* Graph algorithms expressed by a series of sparse matrix operations have clear data-access patterns and can be optimized more easily. Not only can the memory access patterns be optimized for a procedure written as a series of matrix operations, but a PGAS library could exploit this transparency by ordering global communication patterns to hide data-access latencies.

This work represents the first of its kind on this interesting topic of linear algebraic graph algorithms, and represents a collection of original work on the topic that has historically been scattered across the literature. This is an edited volume and each chapter is self-contained and can be read independently. However, the authors and editors have taken great care to unify their notation and terminology to present a coherent work on this topic.

The book is divided into three parts: (I) Algorithms, (II) Data, and (III) Computation. Part I presents the basic mathematical framework for expressing common graph algorithms using linear algebra. Part II provides a number of examples where a linear algebraic approach is used to develop new algorithms for modeling and analyzing graphs. Part III focuses on the sparse matrix computations that underlie a linear algebraic approach to graph algorithms. The book concludes with a discussion of some outstanding questions in the area of large graphs.

While most algorithms are presented in the form of pseudocode, when working code examples are required, these are expressed in MATLAB, and so a familiarity with MATLAB is helpful, but not required.

This book is suitable as the primary book for a class on linear algebraic graph algorithms. This book is also suitable as either the primary or supplemental book for a class on graph algorithms for engineers and scientists outside of the field of computer science. Wherever possible, the examples are drawn from widely known and well-documented algorithms that have already been identified as representing many applications (although the connection to any particular application may require examining the references).

Finally, in recognition of the severe time constraints of professional users, each chapter is mostly self-contained and key terms are redefined as needed. Each chapter has a short summary and references within that chapter are listed at the end of the chapter. This arrangement allows the professional user to pick up and use any particular chapter as needed.

Jeremy Kepner
Cambridge, MA

John Gilbert
Santa Barbara, CA