SIAM REVIEW
Vol. 47, No. 4, pp. 629–705

What Color Is Your Jacobian? Graph Coloring for Computing Derivatives*

Assefaw Hadish Gebremedhin†
Fredrik Manne‡
Alex Pothen†

Abstract. Graph coloring has been employed since the 1980s to efficiently compute sparse Jacobian and Hessian matrices using either finite differences or automatic differentiation. Several coloring problems occur in this context, depending on whether the matrix is a Jacobian or a Hessian, and on the specifics of the computational techniques employed. We consider eight variant vertex coloring problems here. This article begins with a gentle introduction to the problem of computing a sparse Jacobian, followed by an overview of the historical development of the research area. Then we present a unifying framework for the graph models of the variant matrix estimation problems. The framework is based upon the viewpoint that a partition of a matrix into structurally orthogonal groups of columns corresponds to distance-2 coloring an appropriate graph representation. The unified framework helps integrate earlier work and leads to fresh insights; enables the design of more efficient algorithms for many problems; leads to new algorithms for others; and eases the task of building graph models for new problems. We report computational results on two of the coloring problems to support our claims. Most of the methods for these problems treat a column or a row of a matrix as an atomic entity, and partition the columns or rows (or both). A brief review of methods that do not fit these criteria is provided. We also discuss results in discrete mathematics and theoretical computer science that intersect with the topics considered here.

Key words. sparsity, symmetry, Jacobians, Hessians, finite differences, automatic differentiation, matrix partitioning problems, distance-k coloring, approximation algorithms

AMS subject classifications. 05C15, 05C90, 90C06, 90C27, 90C90

DOI. 10.1137/S0036144504444711

When colour goes home into the eyes,
And lights that shine are shut again ...
And that no-place which gave them birth, shall close
The rainbow and the rose ...
—Rupert Brooke, "The Treasure"

*Received by the editors July 1, 2004; accepted for publication (in revised form) April 4, 2005; published electronically October 31, 2005.
http://www.siam.org/journals/sirev/47-4/44471.html

†Computer Science Department, Old Dominion University, Norfolk, VA 23529-0162 (assefaw@cs.odu.edu, pothen@cs.odu.edu). The work of the first author was supported by the University of Bergen while completing his Ph.D. and later by U.S. National Science Foundation grant ACI 0203722. The work of the third author was supported by U.S. National Science Foundation grant ACI 0203722, by DOE grant DE-FC02-01ER25476, by subcontracts B533520 and B542604 from the Lawrence Livermore National Laboratory, and by the Computer Science Research Institute of the Sandia National Laboratories during a sabbatical year.

‡Department of Informatics, University of Bergen, N-5020 Bergen, Norway (fredrikm@ii.uib.no).

Contents.

1. Introduction.

1.1. Overview. Many algorithms that solve nonlinear optimization problems and differential equations require the computation of derivatives. For a vector function, the matrix of first derivatives is the Jacobian; for a scalar function, the matrix of second derivatives is the Hessian. These derivative matrices can be estimated through finite difference approximations or computed exactly, within the limits of machine precision, through automatic differentiation (AD). When the problems are large, many of the elements in the Jacobian or Hessian are zero, and these matrices are said to be sparse. Sparsity and symmetry in the derivative matrices can be exploited to compute the nonzero entries efficiently. In the context of finite difference approximations, efficiency corresponds to reducing the number of function evaluations required. In the context of AD, efficiency often corresponds to reducing the number of AD *passes* required.

The problem of minimizing the number of function evaluations/AD passes needed to compute a sparse derivative matrix can be formulated as a matrix partitioning problem. Indeed, depending on the conditions and the methods employed, one can identify ten variant matrix partitioning problems.

Each of these ten matrix partitioning problems can be modeled as a (specialized) graph coloring problem. The graph coloring models provide deeper insight into the partitioning problems and enable the design of effective approximation or heuristic algorithms. In this paper we revisit the research in this area of the last twenty years with the aim of providing a unifying algorithmic framework. We integrate existing coloring formulations with new ones, design more efficient algorithms for problems that have been studied earlier, and introduce graph models for new matrix computation problems that arise in preconditioning.

1.2. A Jacobian Computation Problem. We begin the discussion of graph coloring models for the various matrix partitioning problems by considering a representative problem, a problem that occurs in a columnwise computation of a Jacobian matrix. After providing a brief background of the problem within the context of a finite difference approach, we will introduce a natural graph coloring formulation.

Consider a nonlinear function $F : R^n \rightarrow R^m$, with $n = m = 5$. Let the nonzero structure of the Jacobian matrix J of the function F be as shown in Figure 1.1(a). In the figure, $j_{k\ell}$ denotes the (k, ℓ) entry of J when it is nonzero; the symbol "0" denotes a zero entry of J. (The illustration in Figure 1.1 is motivated by a figure from Hovland's recent talk [67].)

Let $e_k \in R^n$ correspond to the kth coordinate vector (a vector with 1 in the kth row and 0 in all other rows); then from the approximation $\frac{1}{\epsilon}[F(x + \epsilon e_k) - F(x)] \approx J(x)e_k$, by differencing the function along the direction e_k, we can estimate the kth column of J through the additional function evaluation $F(x + \epsilon e_k)$ (assuming $F(x)$ has already been evaluated). Here ϵ is a small step size. A centered difference formula would be more accurate, but for pedagogical purposes, the forward difference formula suffices. Thus, if sparsity is not exploited, the estimation of a Jacobian matrix with n columns would require n additional function evaluations.

Now consider a subset of the columns of the Jacobian such that no two columns have a nonzero in a common row; such a subset of columns is *structurally orthogonal*. In a group of structurally orthogonal columns, the columns are pairwise orthogonal to each other independent of the numerical values of the nonzeros. In the example, columns 1 and 3 are structurally orthogonal; so are columns 1 and 4, and columns 3 and 5. Choose a column vector d with 1's in components corresponding to the indices of columns in a structurally orthogonal group of columns, and zeros in all

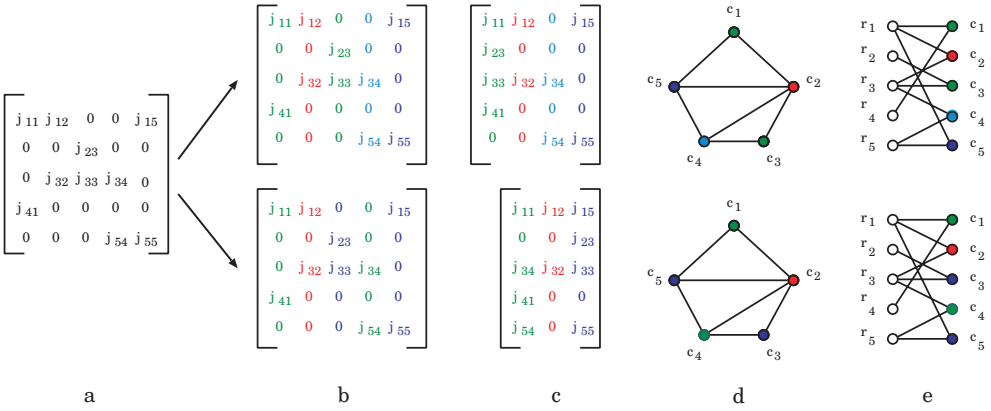

Fig. 1.1 *Two partitions of a Jacobian into groups of structurally orthogonal columns* (a)–(c). *Each partition is also represented as a distance-*1 *coloring in the column intersection graph* (d), *and as a partial distance-*2 *coloring in the bipartite graph* (e). *In the upper row, four colors are used: columns* 1 *and* 3 *are assigned the first color, column* 2 *is assigned the second color, column* 4, *the third color, and column* 5, *the fourth color. In the lower row, three colors are used: columns* 1 *and* 4 *are assigned the first color, column* 2 *is assigned the second color, and columns* 3 *and* 5 *are assigned the third color.*

other components. The vector d is *binary* since each entry is either one or zero. By differencing the function F along the vector d, one can simultaneously determine the nonzero elements in *all* of these columns through one additional function evaluation at $F(x + \epsilon d)$.

Further, by partitioning the columns of the Jacobian into the fewest groups, each consisting of structurally orthogonal columns, the number of (vector) function evaluations needed to estimate the Jacobian matrix is minimized. This leads to the following general problem: given the sparsity structure of a Jacobian matrix, partition the columns into the fewest groups of structurally orthogonal columns.

Figure 1.1(b) shows two different ways in which the Jacobian matrix can be partitioned into groups of structurally orthogonal columns. In the figure, columns of the same color belong to the same group. In the lower row of the figure, the matrix is partitioned into three such groups: $\{1, 4\}$, $\{2\}$, and $\{3, 5\}$. As we will see in section 3.5, the fewest groups of structurally orthogonal columns this matrix can be partitioned into is three, the maximum number of nonzeros in a row.

Two "compressed" representations of the Jacobian matrix in which the nonzeros in a group of structurally orthogonal columns are packed into the same column are shown in Figure 1.1(c). Each column in the compressed representation of the Jacobian corresponds to the nonzeros that can be simultaneously computed through one finite difference operation.

Figure 1.2 shows a Jacobian matrix having 100 columns and its compressed representation with 28 columns obtained by grouping together nonzero elements in a set of structurally orthogonal columns into a single column. This figure is similar to Figures 1.1(b) and (c); the nonzeros within each group of structurally orthogonal columns are assigned a single color distinct from the colors assigned to other nonzeros, and then all nonzeros of the same color are grouped into one compressed column. (Figure 1.2 is inspired by similar figures in Griewank [53] and Hovland [67].)

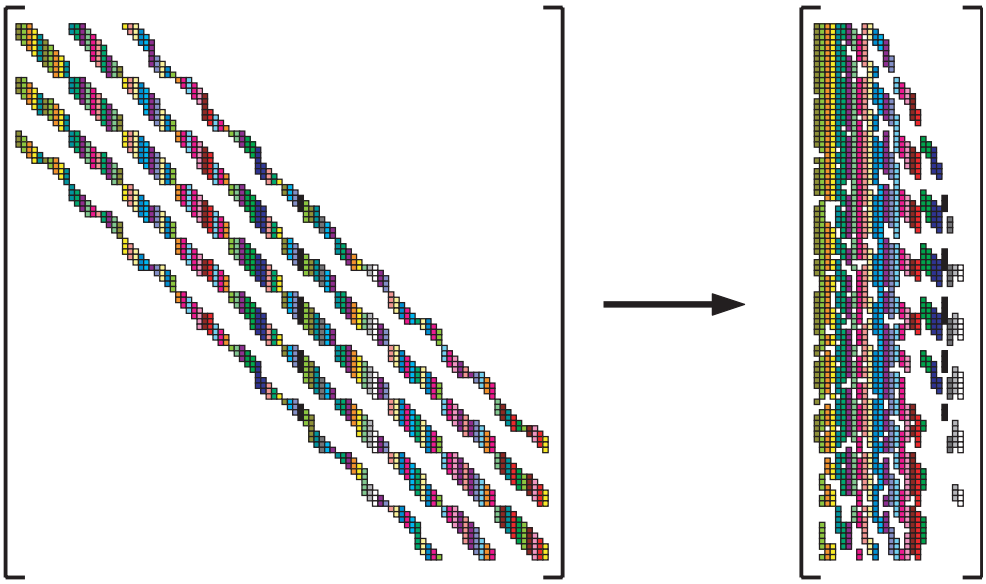

Fig. 1.2 *A Jacobian matrix and its compressed representation.*

Curtis, Powell, and Reid [35] were the first to observe in 1974 that sparsity can be employed in this way to reduce the number of function evaluations needed to estimate the Jacobian. They described a simple greedy algorithm for partitioning the columns of a Jacobian into a small number of groups of structurally orthogonal columns.

In 1983 Coleman and Moré [30] modeled this matrix partitioning problem as a distance-1 graph coloring problem. The model uses the *column intersection graph* of a matrix where columns correspond to vertices and two vertices are joined by an edge whenever the corresponding columns have nonzeros in a common row (i.e., the columns are structurally nonorthogonal). A distance-1 coloring of a column intersection graph, i.e., an assignment of colors to the vertices such that adjacent (distance-1 neighboring) vertices receive different colors, partitions the columns into groups of structurally orthogonal columns. Figure 1.1(d) shows the column intersection graph of the Jacobian in the example and illustrates two different distance-1 colorings of that graph. The lower part of the figure shows that vertices c_1 and c_4 are assigned one color; vertex c_2 is assigned a second color; and vertices c_3 and c_5 are assigned a third color.

The objective in all coloring problems considered in this paper, including the distance-1 coloring problem, is to use the fewest colors possible. Since the distance-1 graph coloring problem is known to be NP-hard, the work of Coleman and Moré showed that it is highly unlikely that there is a polynomial time algorithm for partitioning the columns of a matrix into the fewest groups of structurally orthogonal columns. Meanwhile, they developed several practically effective heuristics for the problem.

In this work, we use a different graph coloring model for the same matrix partitioning problem. This coloring formulation uses a *bipartite graph* to represent a Jacobian matrix. The vertex set V_1 in the bipartite graph corresponds to the rows of the matrix and the vertex set V_2 corresponds to the columns. An edge joins a row vertex r_k to a column vertex c_ℓ if the matrix element $j_{k\ell}$ of the Jacobian is nonzero.

In Figure 1.1, part (e) shows the bipartite graph of the Jacobian whose structure is depicted in part (a).

Notice that two column vertices that are at a distance of two edges from each other in the bipartite graph (hence they share a common row vertex as a neighbor) are structurally nonorthogonal in the matrix. In fact, two columns are structurally orthogonal if and only if they are at a distance greater than 2 from each other in the corresponding bipartite graph. Thus, a distance-2 coloring of the set of column vertices V_2 is equivalent to a partitioning of the columns of the matrix into groups of structurally orthogonal columns. A distance-2 coloring of the vertex set V_2 is an assignment of colors to these vertices such that every pair of column vertices at a distance of exactly two edges from each other receives distinct colors. More precisely, this coloring is a *partial* distance-2 coloring of the bipartite graph since the row vertex set V_1 is left uncolored. Figure 1.1(e) shows two different distance-2 colorings of the column vertices of the bipartite graph of the Jacobian in the example. These colorings are the equivalents of the distance-1 colorings shown in part (d).

In the Jacobian estimation problem under discussion, the bipartite graph–based partial distance-2 coloring formulation may be preferred over the column intersection graph–based distance-1 coloring formulation for several reasons. One of the advantages of the former is that the distance-2 coloring problem serves as the most general problem for the many variations of coloring problems addressed in this paper. The identification of such a general problem simplifies the design of algorithms for the specialized variants. Another advantage is related to the size of the graph used to represent a matrix. The size of the column intersection graph of a matrix A is proportional to the number of nonzeros in A^TA, whereas the size of the bipartite graph of A is proportional to the number of nonzeros in A. For many large-scale practical problems, the former could be considerably larger in size. A larger graph size in turn implies that more storage space is required and performance is degraded due to the hierarchical structure of memory in modern computers [60]. In section 3.6, we provide computational evidence to support this claim. A third advantage of the bipartite graph–based formulation is related to flexibility. Unlike the column intersection graph, the bipartite graph of a matrix is a faithful representation of the sparsity structure and hence allows for a columnwise, rowwise, or combined column- and rowwise computation of a matrix. A detailed discussion of the comparison between the two approaches is included in section 3, a section devoted to the formal treatment of the problem under discussion. In section 9, we consider *hypergraph coloring* formulations for the same problem, where the equivalence between the two graph coloring formulations can be viewed from yet another perspective.

1.3. Automatic Differentiation. For pedagogical reasons, the matrix partitioning problem introduced in section 1.2 has been discussed within the context of a finite difference technique, a technique that computes an approximation to the derivative. The same partitioning problem also arises when a Jacobian matrix is calculated using the relatively recent technique of *automatic differentiation* (AD).

AD is a chain rule–based technique for evaluating the derivatives of functions defined by computer programs. Unlike finite differencing, the derivatives in AD are computed analytically, and hence without any truncation error.

AD has two basic modes of operation known as *forward* and *reverse*. These modes correspond to a bottom-up and a top-down strategy of accumulating partial derivatives of elementary functions that define the computational scheme of the function to be differentiated.

One *pass* of the forward mode of AD can be used to compute a column (or a group of structurally orthogonal columns) of a derivative matrix and a pass of the reverse mode can be used to compute a row (or a group of structurally orthogonal rows). Hence, just as in the finite difference setting, the efficient computation of a Jacobian matrix using the forward mode of AD requires the columns to be partitioned into as few groups of structurally orthogonal columns as possible. Similarly, an efficient application of the reverse mode requires a row-partitioning into the fewest possible groups of structurally orthogonal rows.

A number of AD software tools are available, including ADIFOR [16], Odyssee [110], and TAMC [50] for FORTRAN 77, and ADIC [17] and ADOL-C [55] for C/C++.

Some AD tools, such as ADIFOR and ADIC, support a *vector mode* in which several bidirectional derivatives are computed simultaneously. Using such tools, the compressed derivative matrix (see Figure 1.2) can be computed in one pass. The computational effort involved in these contexts is proportional to the number of columns in the compressed matrix. Hence a partition into the fewest groups of structurally orthogonal columns continues to capture the requirement of an efficient computation.

1.4. Variations on Matrix Computation. As mentioned earlier, depending on the type of derivative matrix being computed and the specifics of the method being applied, there exist several variant matrix partitioning problems. Specifically, the nature of a particular problem in our context depends on the following:

- whether the matrix to be computed is a *Jacobian* (nonsymmetric) or a *Hessian* (symmetric);
- whether the evaluation scheme employed is *direct* or *substitution*-based (a direct method requires solving a diagonal system and a substitution method relies on solving a triangular system of equations);
- whether a *unidirectional* (1d) partition or a *bidirectional* (2d) partition is used (a unidirectional partition involves only columns or rows, whereas a bidirectional one involves both columns and rows);
- whether *all* of the nonzero entries of the matrix or only a *subset* need to be determined; we refer to these as *full* and *partial* matrix computation.

Under certain assumptions, the aforementioned four mutually orthogonal factors give rise to ten variants of matrix partitioning problems. Figure 1.3 depicts a classification of the problems using these factors; the first eight of the ten listed problems are studied in this paper, the problems being numbered in the order in which they appear in the body of this paper.

Each of these matrix partitioning problems can be modeled as a graph coloring problem. In section 1.2 we have seen the graph coloring formulations of problem P1, a partitioning problem arising in the computation of all nonzero entries of a Jacobian matrix using a unidirectional partition via a direct method. In sections 1.4.1 through 1.4.4, we briefly review the historical development of the coloring formulations for problems P2 through P5 and introduce the new problems P6 through P8. The discussion is organized along the four factors underlying the classification in Figure 1.3. Problems P1 through P8 are formally treated in detail in sections 3 through 8.

Note that the classification in Figure 1.3 is oblivious as to whether the chosen numerical technique is finite difference or AD. For example, as has been pointed out earlier, problem P1 arises within the context of finite difference or AD. In our discussion in what follows, we may refer to one numerical technique that simplifies the presentation, but the reader should bear in mind that the particular partitioning problem also arises within the context of the other numerical technique.

Full					Partial				
1d partition			2d partition		1d partition			2d partition	
Direct		Substi-tution	Direct	Substi-tution	Direct		Substi-tution	Direct	Substi-tution
Jac. P1	Hes. P2	Hes. P4	Jac. P3	Jac. P5	Jac. P6	Hes. P7	Hes. P9	Jac. P8	Jac. P10

Fig. 1.3 *Classification of partitioning/coloring problems arising in sparse derivative matrix computation. Problems* P1 *through* P8 *are addressed in this paper.*

1.4.1. Jacobian vs. Hessian. In 1979 Powell and Toint [107] extended the approach of Curtis, Powell, and Reid to compute sparse Hessians. Their approach exploits the fact that only one of each pair of symmetric off-diagonal elements in the Hessian matrix needs to be computed.

McCormick [97] introduced a distance-2 graph coloring model for the computation of Hessians in 1983. The model uses the *adjacency graph* representation of the underlying symmetric matrix and requires that in every path u, v, w in the graph, vertices u, v, and w receive distinct colors. Independently, in 1984, Coleman and Moré [31] gave a more precise coloring model that exploits symmetry. They called the model *path coloring* and required it to satisfy the two conditions: (1) every pair of adjacent vertices receives distinct colors (a distance-1 coloring), and (2) every path on four vertices uses at least three colors. This variant of coloring was called *star* coloring, e.g., by Fertin, Raspaud, and Reed [41], since in such a coloring every subgraph induced by vertices assigned any two colors is a collection of stars. We will use the name star coloring in this paper since path coloring is a term used in the optical networks literature to refer to a different concept.

Unlike coloring problems corresponding to general matrices, coloring problems in discretization of *structured* grids can be solved optimally in polynomial time. In 1984, Goldfarb and Toint [52] studied such optimal estimation of Jacobians and Hessians that arise as model problems in finite difference approximations of partial differential equations.

1.4.2. Direct vs. Substitution-Based Evaluation. The evaluation scheme considered thus far is a direct one, in that each nonzero element of a derivative matrix is obtained from some row of a matrix-vector product via a finite difference operation or an AD pass. Suppose A is an $m \times n$ derivative matrix and $d_1, d_2, \ldots, d_p$ are binary vectors (directions) corresponding to a column partition of A into p groups of structurally orthogonal columns. In the AD literature, the $n \times p$ matrix D consisting of the p columns $d_1, d_2, \ldots, d_p$ is known as a *seed matrix*. In a direct evaluation scheme, the product AD can be used to define a *diagonal* system of equations where the unknowns are the nonzero entries of A and the right-hand side is the result obtained by p function evaluations or p passes of the forward mode of AD.

An alternative evaluation scheme would be substitution-based, in which the unknown matrix elements were determined by solving a triangular system of equations.

The latter is obtained by an appropriate (less restricted) choice of a seed matrix that partitions the columns of the derivative matrix into groups of not necessarily structurally orthogonal columns. There is a trade-off in the choice between a direct and a substitution method. The former being more restrictive requires more function evaluations (or AD passes), while the latter is subject to numerical instability. Moreover, unlike a direct method where the unknowns are obtained without any further arithmetic, a substitution method incurs the additional computational cost involved with solving a sparse triangular system of equations.

A substitution-based evaluation is often effectively combined with the exploitation of symmetry and hence is used in computing the Hessian. A substitution method for computing the Hessian leads to an *acyclic coloring* problem in which the requirements are that (1) the coloring corresponds to a distance-1 coloring, and (2) vertices in every cycle of the graph are assigned at least three distinct colors. This variant of coloring is called acyclic since every subgraph induced by vertices assigned any two colors is a collection of trees—and hence is acyclic. Substitution methods for computing Hessians were studied by Powell and Toint [107] and Coleman and Moré [31]. The latter authors in particular found a less accurate model (in comparison with acyclic coloring) called *triangular coloring*. The acyclic coloring formulation is due to Coleman and Cai [27], who called the coloring variant *cyclic*. The name acyclic coloring was introduced by Grünbaum [56], the first to define and study the problem within an entirely different context.

A variant of a substitution method for computing a Jacobian matrix has been considered by Hossain and Steihaug [63]. The method relies on first finding a partition of the columns into groups of structurally orthogonal columns and then merging a pair of successive groups to get a column grouping that allows overlaps; in this way a column *partition* consisting of p groups is changed into a *grouping* having $p-1$ groups.

1.4.3. Unidirectional vs. Bidirectional Partition. The third source for problem variation while computing the Jacobian lies in one's choice to compute by partitioning the columns (as we have discussed earlier), or rows, or both columns and rows. If the Jacobian has a few dense columns, i.e., columns with a large number of nonzeros, it may be more advantageous to consider a partitioning of the columns. Similarly, if the matrix has a few dense rows, a partitioning of the rows may pay off. However, if the matrix contains a few dense columns *and* rows, it may be better to consider partitioning subsets of *both* columns and rows. A partition that involves only columns or rows is referred to as *unidirectional*, and one that involves both columns and rows is called *bidirectional*. More specifically, given the structure of an $m \times n$ Jacobian matrix A, a bidirectional partitioning problem is concerned with finding seed matrices $D_1 \in \{0,1\}^{n \times p_1}$ and $D_2 \in \{0,1\}^{m \times p_2}$ such that the products AD_1 and $D_2^T A$ together enable the determination of the entries of A and the value $p = p_1 + p_2$ is minimized. Due to symmetry, there is no advantage in considering a bidirectional partition of the Hessian; i.e., a symmetry-exploiting unidirectional partition suffices.

In the context of AD, bidirectional partitions arise when the Jacobian is computed by using the forward and reverse modes simultaneously.

Bidirectional partitioning of the Jacobian leads to specialized *bicoloring* problems in the bipartite graph, i.e., a coloring of subsets of both the row vertices and the column vertices with disjoint sets of colors. When bidirectional partitioning is used within a direct evaluation scheme for Jacobians, the coloring problem is that of *star bicoloring*; the corresponding model within a substitution-based scheme is the *acyclic bicoloring* problem. Bidirectional partitioning problems and their graph

Table 1.1 *Graph coloring formulations for computing* all *nonzero entries of derivative matrices. The Jacobian is represented by its bipartite graph, and the Hessian by its adjacency graph. NA stands for not applicable. The labels in parentheses correspond to those used in Figure* 1.3.

	1d partition	2d partition	
Jacobian	Distance-2 coloring [P1]	Star bicoloring [P3]	Direct
Hessian	Star coloring [P2]	NA	Direct
Jacobian	NA	Acyclic bicoloring [P5]	Substitution
Hessian	Acyclic coloring [P4]	NA	Substitution

coloring formulations were studied by Hossain and Steihaug [62] and Coleman and Verma [32].

1.4.4. Full vs. Partial Computation. The final variation within the classification scheme of Figure 1.3 is whether all elements of the Jacobian and the Hessian are required, or only a subset that is needed for preconditioning purposes. We refer to these variations as *full* and *partial* matrix computation. The latter would be useful in "matrix-free" methods for large-scale problems, where the Jacobian is too large to be explicitly estimated, but a coarser representation of the Jacobian is used as a preconditioner. Partial matrix computation problems lead to restricted coloring problems where only a specified subset of the vertices need to be colored; however, one still needs to pay attention to the remaining vertices, since they could interfere with the estimation of the required matrix elements. Partial matrix computation problems and their coloring formulations are treated in this paper for the first time.

All of these variations lead to a rich collection of graph coloring problems. Table 1.1 shows the collection of five coloring problems that arise when we consider the computation of all nonzero entries of Jacobians and Hessians. Partial matrix computation problems lead to another set of five coloring problems, of which we have formulated graph models only for direct methods, i.e., problems P6 through P8 in Figure 1.3. The precise formulation of each of these coloring problems is described in the section of the paper devoted to the corresponding problems.

1.4.5. Other Variations. One common feature in the partitioning/coloring problems cataloged in Figure 1.3 is that the nonzero entries of a derivative matrix are obtained by solving either a diagonal or a triangular system of equations. A second common feature is that the system of equations is produced from a *partition* of a set of columns or rows (or both) of a matrix; i.e., a set is divided into groups in which an element of the set belongs to one and only one group. Newsam and Ramsdell [103] suggested a method for computing a Jacobian that relies on solving an overdetermined *rectangular* system of equations, a system produced from a column grouping that is not necessarily a partition. The approach of Newsam and Ramsdell has been applied within the context of AD by Geitner, Utke, and Griewank [49]. Recently, Hossain and Steihaug [64] suggested a method for computing a Jacobian that requires solving a banded rectangular system of equations.

A third common feature in the problems of Figure 1.3 is that the granularity level used in defining the problems is an entire column or row of a matrix. In other words, a column or a row is not divided any further. However, approaches that rely on more "fine-grained" partitioning have also been suggested. The *element isolation* method of Newsam and Ramsdell [103] and the *segmented column* method of Hossain and Steihaug [61, 65] are examples of such approaches.

The works mentioned in the previous two paragraphs are briefly discussed in section 10 of this article.

1.5. Objectives. Since the work of Curtis, Powell, and Reid [35], structural orthogonality in a sparse derivative matrix has been identified as a basic property to be exploited for efficient computation. Two columns of a Jacobian matrix are structurally orthogonal if and only if they are at a distance greater than two in the bipartite graph of the Jacobian. Similarly, two columns of a Hessian matrix are structurally orthogonal if and only if they are at a distance greater than two in the adjacency graph of the Hessian. Thus distance-2 coloring of the bipartite graph of the Jacobian and the adjacency graph of the Hessian are the two archetypal problems in efficient derivative matrix computation. This perspective emphasizes the central role played by distance-2 coloring and thus provides a unifying algorithmic framework for the many coloring problems.

Describing the unified algorithmic framework is the first objective of this paper; this framework simplifies the development of algorithms and software for the problems that have been considered in earlier work and clarifies the development of models and algorithms for the new partial Jacobian and Hessian computation problems introduced here. Many of the algorithms proposed earlier for the variant problems relied on adding new edges to the graph until a distance-1 coloring of the transformed graph could be used to solve the problem. We show that more efficient algorithms are possible for the variant problems by employing coloring algorithms directly to the natural graph representations of the problem.

A second objective of this paper is to provide a self-contained introduction to the use of graph coloring in derivative matrix computations, in order to make this work accessible to a broad audience. For researchers in optimization, this article provides the first review of graph coloring methods for efficiently computing derivatives for nonlinear problems. The article has been organized by the manner in which these computational methods would be used in optimization: Jacobian or Hessian matrix; direct or substitution method; unidirectional or bidirectional estimation; full matrix or partial matrix evaluation. Our hope is that this organization makes it easy for practitioners to read the subset of topics of interest to them. An alternate way of organizing this material would have been by the graph algorithms employed to solve the problems, and indeed, that is how we first attempted to present this material.

Work done in other research communities on graph coloring topics intersect with the problems in derivative matrix computation. For instance, early results obtained by the graph theory community on vertex orderings for reducing the number of colors have been applied in the optimization context. The graph theory community has also recently considered some of the variant coloring problems discussed here; e.g., star coloring, which arises in Hessian estimation [5, 41, 102]. However, this community seems to be unaware of work done nearly twenty years earlier on this problem in optimization! Conversely, earlier work on acyclic coloring was not noticed by researchers in optimization who discovered its application to evaluating derivative matrices with substitution methods.

Graph and hypergraph coloring have been used in a wide collection of application areas in addition to optimization: register allocation in compilers [24], radio and wireless networks [44, 84], scientific computing [51, 73, 74, 111], data movement in distributed and parallel computing [58], facility location problems [117], cache-efficient algorithms [77], etc. Parallel computers make it feasible to solve large-scale problems in many of these application areas, especially optimization, and hence there is cur-

rently increased interest in efficient algorithms and software for coloring graphs with millions of vertices.

For the reasons spelled out in the previous two paragraphs, wider dissemination of this work among researchers in the optimization, scientific computing, computer science, discrete mathematics, and various applications communities would be beneficial.

A third objective of this work is to help dispel the feeling that most of the interesting work on this topic has been done already by pointing to the research frontier in this area: new coloring and ordering algorithms need to be developed and implemented for many problems considered in this article; new mathematical results that characterize lower bounds for coloring special classes of graphs are needed. We believe that as this work becomes better known, new applications of coloring will be discovered, further enriching the study of the mathematics and algorithms in this area.

1.6. New Contributions. This article gives a coherent review of earlier work, but it also has several new results. The following summary highlights our contributions.

1. We propose distance-2 coloring as a natural, flexible, space-efficient model for problem P1 (section 3).
2. We expose the inter-relationships among the coloring formulations of the various matrix partitioning problems and identify distance-2 coloring as a unifying, archetypal model (section 7, building on the discussions in sections 3–6).
3. Using the insight gained from the unified graph-theoretic treatment, we develop several greedy heuristic algorithms, including new algorithms for star coloring (section 4) and star bicoloring (section 5). We also sketch a new acyclic coloring algorithm that exploits the structure of two-colored induced subgraphs (section 6). In a separate paper (see [48] for a draft), we will describe this algorithm in more detail, as well as a new star coloring algorithm based on the same paradigm.
4. We report experimental results from our implementations that demonstrate
   - the advantages offered by the bipartite graph–based distance-2 coloring formulation of problem P1 in comparison with the column intersection graph–based distance-1 coloring formulation (section 3), and
   - the time/quality trade-off between two star coloring algorithms; one of the algorithms is new while the other is a translation from a previously known matrix-based algorithm (section 4).
5. In the context of bicoloring models, we make the connection to the problem of finding a *vertex cover* in a graph explicit (section 5).
6. We develop the first graph coloring formulations for partial matrix computation problems, problems where only a subset of the nonzero entries of a matrix is required to be computed for preconditioning purposes (section 8).
7. We give the first *hypergraph* coloring formulations for problem P1. These formulations provide an interesting alternative perspective and are likely to be useful in the context of partial Jacobian and Hessian computation (section 9).
8. We formulate the *element isolation* technique of Newsam and Ramsdell [103] for computing a Jacobian as an *edge coloring* problem in the associated bipartite graph (section 10).

1.7. Scope and Further Reading.

Automatic Differentiation. A discussion of the technical details of AD is beyond the scope of this paper. The book by Nocedal and Wright [104] contains an introductory discussion of AD and how graph coloring can be used to reduce the number of passes

Table 1.2 *A list of concepts known by different names. The names acyclic and star coloring were earlier used by Grünbaum* [56] *and Fertin, Raspaud, and Reed* [41]*, respectively.*

Here	Elsewhere
Star coloring	Symmetric coloring [31]
	Path coloring [27]
	Distance-$\frac{3}{2}$ coloring [45, 47]
Acyclic coloring	Cyclic coloring [27]
Star bicoloring	Path bicoloring [32]
Acyclic bicoloring	Cyclic bicoloring [32]
Structurally orthogonal partition	Consistent partition [30]
Symmetrically orthogonal partition	Symmetrically consistent partition [31]
Bidirectional partition	Bipartition [32]
	Row-column partition [62]

needed to compute the Jacobian and the Hessian. The recent book by Griewank [53] has an excellent discussion of the principles and techniques of AD. The books [33, 54] contain several papers that discuss different aspects of AD, including its theory, implementations, and applications. Examples of works that discuss AD within the context of parallel computation include [68, 69, 106].

Combinatorial problems other than coloring also occur in optimizing the computation of derivatives using AD. A graph model for *eliminating* vertices and edges in a computational graph can be used in this context to reduce the storage and the effort needed to compute the Jacobian or the Hessian. Depending on the type of elimination technique employed, there exist a number of combinatorial problems that could be regarded as searching for shortest paths in graphs. These problems are similar to graph models of sparse Gaussian elimination. Naumann [101] has recently studied graph elimination in AD. Examples of studies that investigate such methods can also be found in [33, 53, 54, 100]. In a recent talk, Hovland [67] surveyed the various combinatorial problems that occur in AD. We will not discuss these problems in this paper.

Graph Coloring. The graph coloring literature per se is vast. This work focuses on coloring problems that arise in computing derivative matrices. Furthermore, our emphasis is on problem formulations, graph models, and fast heuristic or approximation algorithms. Iterative local improvement heuristics and exact algorithms for the coloring problems are not discussed. In section 11 we include a discussion of effective vertex orderings for greedy distance-1 coloring, a discussion of the *coloring number* of a graph, and a brief review of theoretical results on some of the coloring problems addressed in this paper. The monograph by Jensen and Toft [71] provides a detailed account of over 200 open coloring problems and an extensive list of references. The survey article by Toft [115] has a good discussion of mathematical results for the distance-1 coloring problem. The recent book edited by Kubale [86] includes survey articles on several different coloring problems. The WWW pages of Culberson [34] and Trick [116] are two useful web resources for the distance-1 coloring problem.

1.8. Terminology. For some of the concepts described in this paper, we have introduced names that differ from those used earlier in the numerical optimization literature. We have done so when we felt that the new names are more descriptive or consistent with the terminology in other fields. For easy reference, Table 1.2 lists the important concepts in this paper that have been named differently elsewhere.

2. Definitions of Graph-Theoretic Concepts. In the current section we define most of the graph-theoretic concepts used throughout this paper. Other concepts will be defined later as required.

2.1. Preliminary Concepts and Notations. A *graph* G is an ordered pair (V, E), where V is a finite and nonempty set of *vertices* and E is a set of unordered pairs of distinct vertices called *edges*. If $(u, v) \in E$, vertices u and v are said to be *adjacent*; otherwise they are called *nonadjacent*. A *path* of *length* ℓ (edges) in a graph is a sequence $v_1, v_2, \ldots, v_{\ell+1}$ of distinct vertices such that v_i is adjacent to v_{i+1} for $1 \leq i \leq \ell$. Two distinct vertices are said to be *distance-k neighbors* if a shortest path connecting them has length *at most* k. Note that the set of distance-k neighbors of vertex u, denoted by $N_k(u)$, does not include u itself. Note also that, by definition, two distance-k neighboring vertices are also distance-k' neighbors for $k' > k$. The usual definition of the *degree* of a vertex is the number of edges incident on it. We extend this notion and define the *degree-k* of a vertex u, denoted by $d_k(u)$, to be the number of distinct paths of length at most k edges starting at u. The terms degree and degree-1 are thus synonymous, and they are used interchangeably in this paper. The average degree-k in a graph is denoted by $\overline{\delta}_k$. For brevity, we often write $d(u)$ instead of $d_1(u)$, and $\overline{\delta}$ instead of $\overline{\delta}_1$. The maximum degree-1 in a graph is (consistently) denoted by Δ, and the minimum degree-1 is denoted by δ.

For a simple example consider the graph shown on the left-hand side of Figure 2.1. In this example, the distance-1 and distance-2 neighbors of vertex a are $N_1(a) = \{b, c\}$ and $N_2(a) = \{b, c, d, e\}$; the degree of vertex c is $d_1(c) = 4$, which is also the maximum degree, Δ, in the graph. The degree-2 of the vertex c is $d_2(c) = 8$.

A graph is called *bipartite* if its vertex set can be partitioned into two disjoint sets V_1 and V_2 such that every edge of the graph connects a vertex in V_1 with a vertex in V_2. In a bipartite graph, we denote the maximum degree in the vertex set V_1 by $\Delta(V_1)$, and the maximum degree in the vertex set V_2 by $\Delta(V_2)$. Similarly, the average degree-k in the sets V_1 and V_2 are denoted by $\overline{\delta}_k(V_1)$ and $\overline{\delta}_k(V_2)$, respectively. The graph in Figure 1.1(e) is an example of a bipartite graph; V_1 consists of the row-vertices and V_2 consists of the column-vertices. In this graph, $\Delta(V_1) = 3$ and $\Delta(V_2) = 2$.

A graph is *connected* if any two of its vertices are linked by a path in the graph. A *tree* is a connected graph without any cycle. A *forest* is a graph each of whose components is a tree. A bipartite graph $G_b = (V_1, V_2, E)$ in which each vertex in V_1 is connected to every vertex in V_2 is *complete*. A complete bipartite graph in which one of the vertex sets consists of a single vertex is a *star*.

Let $G = (V, E)$ be a graph. A set of vertices $C \subseteq V$ is said to *cover* a set of edges $F \subseteq E$ if for every edge $e \in F$, at least one of the endpoints of e is in C. If the vertex set C covers the set of edges E, it is called a *vertex cover*. A set of vertices $I \subseteq V$ is called an *independent set* if no two vertices in I are adjacent to each other. A set of vertices $Q \subseteq V$ is called a *clique* if the vertices in Q are pairwise mutually adjacent to each other. For any set of vertices $U \subseteq V$, the graph *induced* by U is the subgraph of G whose vertex set is U and whose edges are precisely the edges of G with both endpoints in U. The graph induced by U is denoted by $G[U]$.

In the graph on the left in Figure 2.1, the set $\{a, c, f\}$ is a vertex cover, the set $\{a, d, e\}$ is an independent set, and the set $\{a, b, c\}$ is a clique.

2.2. Distance-*k* Graph Coloring. A *distance-k* (vertex) coloring of a graph $G = (V, E)$ is a mapping $\phi : V \to \{1, 2, \ldots, p\}$ such that $\phi(u) \neq \phi(v)$ whenever vertices u and v are distance-k neighbors. The color assignment to the vertices of the graph on the left in Figure 2.1 shows an example of a distance-2 coloring. The least possible number of colors required for a distance-k coloring of a graph G is called its *k-chromatic number* and is denoted by $\chi_k(G)$. A distance-k coloring of $G = (V, E)$ is called *partial* if only a subset of the vertices is colored; in particular, a partial

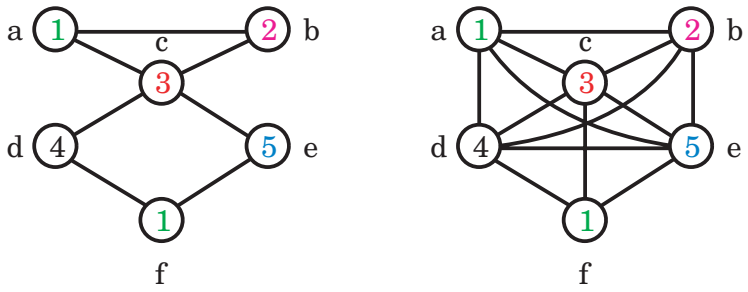

Fig. 2.1 *An example of a graph G (left) and its square graph G^2 (right). The color assignments (numbers within the circles) show a distance-2 coloring of G and the equivalent distance-1 coloring of G^2.*

distance-k coloring of $G = (V, E)$ on W, $W \subset V$, is a mapping $\phi : W \to \{1, 2, \ldots, p\}$ such that $\phi(u) \neq \phi(v)$ whenever vertices u and v in the set W are distance-k neighbors. Figure 1.1(e) shows two examples of a partial distance-2 coloring of a bipartite graph on one of its vertex sets.

2.3. The Power of a Graph. The *kth power* of a graph G is the graph G^k whose vertex set is the same as that of G and whose edge set consists of pairs of vertices (u, v) whenever vertices u and v are distance-k neighbors in G. For example, the graph on the right in Figure 2.1 is the square graph G^2 of the graph G shown on the left. The following equivalence is easy to see.

LEMMA 2.1. *A mapping ϕ is a distance-k coloring of G if and only if it is a distance-1 coloring of G^k.*

Figure 2.1 illustrates the equivalence between a distance-2 coloring of a graph and a distance-1 coloring of its square.

2.4. Representing Matrices Using Graphs. Here we will define different graph representations of the sparsity structure of matrices. First, a remark on notation is in order. In the rest of this paper, for a given matrix A, the ith row is denoted by r_i, the jth column is denoted by a_j, and the (i, j) entry is denoted by a_{ij}. Moreover, in order to simplify notation, we use r_i (a_j) to refer both to the ith row (jth column) of matrix A and to the corresponding vertex in an appropriate graph representation of A.

Bipartite Graph. Let A be an $m \times n$ matrix with rows $r_1, r_2, \ldots, r_m$ and columns $a_1, a_2, \ldots, a_n$. The bipartite graph $G_b(A)$ of A is defined as $G_b(A) = (V_1, V_2, E)$, where $V_1 = \{r_1, r_2, \ldots, r_m\}$, $V_2 = \{a_1, a_2, \ldots, a_n\}$, and $(r_i, a_j) \in E$ whenever a_{ij} is nonzero for $1 \leq i \leq m$, $1 \leq j \leq n$. For example, Figure 1.1(e) shows the bipartite graph representation of the matrix whose nonzero structure is shown in part (a).

The size of the graph $G_b(A)$ is proportional to the size of the matrix A: the number of vertices $|V_1| + |V_2| = m + n$, and the number of edges $|E| = nnz(A)$, where $nnz(A)$ is the number of nonzeros in A.

Adjacency Graph. Let A be an $n \times n$ symmetric matrix having nonzero diagonal elements with its columns denoted $a_1, a_2, \ldots, a_n$. The adjacency graph of A is $G(A) = (V, E)$, where $V = \{a_1, a_2, \ldots, a_n\}$, and $(a_i, a_j) \in E$ whenever a_{ij} is nonzero for $1 \leq i \leq n$, $1 \leq j \leq n$, $i \neq j$.

The graph $G(A)$ exploits the symmetry available in A since both nonzero entries a_{ij} and a_{ji} are represented by the single edge (a_i, a_j). In the adjacency graph $G(A)$,

the edges corresponding to the nonzero diagonal elements of A are not explicitly represented. Thus, the number of edges in $G(A)$ is $\frac{1}{2}(nnz(A) - n)$, where $nnz(A)$ is the number of nonzeros in A. Since there is a vertex corresponding to each column, the number of vertices in $G(A)$ is n.

Column Intersection Graph. The graph formulations for the derivative matrix computation problems considered in this paper are based solely on the two graph representations discussed above. However, the column intersection graph has also been used to represent a nonsymmetric matrix.

Let $a_1, a_2, \ldots, a_n$ now correspond to the *structures* of the columns of an $m \times n$ matrix A. In particular, let each $a_j = \{i \in \{1, \ldots, m\} : a_{ij} \neq 0\}$ (i.e., a_j is a set of row indices in which the jth column of A has a nonzero entry). The column intersection graph of A is $G_c = (V, E)$, where $V = \{a_1, \ldots, a_n\}$ and $(a_i, a_j) \in E \iff a_i \bigcap a_j \neq \emptyset$. In other words, an edge (a_i, a_j) exists whenever both the ith and the jth column of A have nonzero entries in at least one common row index. For example, Figure 1.1(d) shows the column intersection graph of the matrix whose nonzero structure is shown in part (a).

Note that the column intersection graph of a matrix A is isomorphic to the adjacency graph of $A^T A$.

3. Unidirectional, Direct Computation of the Jacobian. In this section we revisit the partitioning problem introduced in section 1.2 with more technical details.

Given a continuously differentiable function $F : R^n \to R^m$, the *Jacobian* of F at the point x is the $m \times n$ matrix whose (i, j) entry $F'(x)_{ij} = \frac{\partial f_i}{\partial x_j}(x)$, where $f_1(x), f_2(x), \ldots, f_m(x)$ are the components of $F(x)$. Let A denote the Jacobian matrix $F'(x)$. An estimate for the jth column of A can be obtained from the finite difference approximation

$$Ae_j = a_j = \frac{\partial}{\partial x_j} F(x) \approx \frac{1}{\epsilon}[F(x + \epsilon e_j) - F(x)], \qquad 1 \leq j \leq n, \tag{3.1}$$

where e_j is the jth coordinate vector and ϵ is a positive step length. Formula (3.1) shows that if each column of A is computed independently, n function evaluations, in addition to the evaluation of $F(x)$, will be required. However, by exploiting the sparsity structure of A, the required number of function evaluations can be reduced significantly. Often the sparsity structure of A is either available or can be obtained relatively easily [53, 61]. The goal here is to exploit the known sparsity structure of A to estimate its nonzero entries using as few function evaluations as possible. We assume that evaluating the function F at a given point is more efficient than evaluating the components f_i, for $1 \leq i \leq m$, separately. This assumption is reasonable since in many applications the components of F have common subexpressions, and hence the computation can be made efficient by avoiding repeated computation of these subexpressions.

A sparsity-exploiting estimation of a matrix using finite differences involves finding directions (vectors) $d_1, d_2, \ldots, d_p$ such that the products $Ad_1, Ad_2, \ldots, Ad_p$ enable the determination of all the nonzero entries of A. Here each vector d is *binary* (i.e., each entry is either 1 or 0) and is obtained via some linear combination of the coordinate vectors. Using (3.1), the products $Ad_1, Ad_2, \ldots, Ad_p$ can be used to define a system of linear equations in which the unknowns are the nonzero elements of A. If the choice of the binary vectors d_i is such that the resulting system of equations can be ordered to be diagonal, then A can be determined *directly*. If the vectors d_i are chosen such that the system of equations can be ordered to be triangular, then the unknowns can be determined via *substitution*.

In the current and the next two sections, we consider direct methods; substitution methods will be discussed in section 6.

3.1. The Matrix Partitioning Problem. In a direct method for determining a Jacobian matrix A, for each nonzero element a_{ij} there should exist some vector d_k in the set $\{d_1, d_2, \ldots, d_p\}$ such that $a_{ij} = (Ad_k)_i$. Here $(Ad_k)_i$ denotes the ith entry of the vector Ad_k. Thus each nonzero matrix element a_{ij} can be read off from an entry of some vector Ad_k.

In an efficient estimation of a Jacobian, the number of vectors p, which is proportional to the number of function evaluations required, needs to be minimized. Hence the following general problem can be stated.

PROBLEM 3.1. *Given the sparsity structure of an $m \times n$ matrix A, find the fewest binary vectors $d_1, d_2, \ldots, d_p$ such that the products $Ad_1, Ad_2, \ldots, Ad_p$ enable a direct determination of A.*

Curtis, Powell, and Reid [35] were the first to address Problem 3.1. They observed that the vectors $d_1, d_2, \ldots, d_p$ can be obtained by partitioning the columns of A into groups of structurally orthogonal columns.

DEFINITION 3.2. *A partition of the columns of a matrix A is* structurally orthogonal *if for every nonzero element a_{ij} the group containing column a_j has no other column with a nonzero in row r_i.*

Let $\{C_1, C_2, \ldots, C_p\}$ be a structurally orthogonal column partition. With each group C_k, associate a binary vector d_k having components $\delta_j = 1$ if a_j belongs to C_k, and $\delta_j = 0$ otherwise. Then

$$Ad_k = \sum_{j=1}^{n} \delta_j a_j = \sum_{a_j \in C_k} a_j.$$

If $a_{ij} \neq 0$ and column $a_j \in C_k$, then $a_{ij} = (Ad_k)_i$. Thus, all the nonzero entries of A can be determined from the p matrix-vector products $Ad_1, \ldots, Ad_p$.

Problem 3.1 can thus be cast as a matrix partitioning problem in the following way.

PROBLEM 3.3. *Given the sparsity structure of an $m \times n$ matrix A, find a structurally orthogonal partition of its columns that has the fewest groups.*

As discussed in the introduction, Problem 3.3 also arises while computing the derivative matrix A using AD. In such a context, the number of groups in a structurally orthogonal partition corresponds to the number of passes of the forward mode of AD.

3.2. A Graph Coloring Formulation. Recall that a distance-k coloring of a graph $G = (V, E)$ is a mapping $\phi : V \to \{1, 2, \ldots, p\}$ such that $\phi(u) \neq \phi(v)$ whenever u and v are distance-k neighbors. The distance-k graph coloring problem asks for a distance-k coloring with the fewest colors.

A distance-k coloring of $G = (V, E)$ that uses p colors partitions the vertex set V into p *color classes* $U_1, U_2, \ldots, U_p$, where $U_i = \{u \in V : \phi(u) = i\}$. Each color class is a *distance-k independent set*; i.e., no pair of distinct vertices in the class consists of distance-k neighbors. This prompts the question, Is there a natural relationship between Problem 3.3 and the distance-k graph coloring problem? The following simple observation gives a graph-theoretic characterization of structural orthogonality in a nonsymmetric matrix, which leads to an answer to this question.

LEMMA 3.4. *Let A be an $m \times n$ matrix and $G_b(A) = (V_1, V_2, E)$ be its bipartite graph. Two columns (or rows) in A are structurally orthogonal if and only if the corresponding vertices in $G_b(A)$ are at a distance greater than two from each other.*

Proof. We prove the statement for columns; a similar argument can be used to prove the case for rows. First notice that in the bipartite graph $G_b(A)$, any path between a pair of distinct column vertices is of even length.

Assume that vertices a_i and a_j in V_2 are at a distance greater than two from each other. This implies that there exists no path a_i, r_k, a_j in G_b for any $r_k \in V_1$, $1 \leq k \leq m$. In terms of matrix A, this means that there is no $k \in [1, m]$ such that both a_{ki} and a_{kj} are nonzero. Hence, by definition, columns a_i and a_j are structurally orthogonal.

Conversely, assume that columns a_i and a_j are structurally orthogonal. Then, by definition, there is no $k \in [1, m]$ such that $a_{ki} \neq 0$ and $a_{kj} \neq 0$. This implies that there is no path a_i, r_k, a_j in $G_b(A)$ for any $1 \leq k \leq m$. Hence, vertices a_i and a_j are at a distance $d > 2$ from each other. In particular, vertices a_i and a_j are a distance $d \geq 4$ apart. □

By Lemma 3.4, finding a structurally orthogonal partition of the columns of a matrix A is equivalent to finding a *partial* distance-2 coloring of $G_b(A) = (V_1, V_2, E)$ on V_2. (The coloring is partial since V_1 is not colored.) Theorem 3.5 states this equivalence.

THEOREM 3.5. *Let A be an $m \times n$ matrix and $G_b(A) = (V_1, V_2, E)$ be its bipartite graph representation. A mapping ϕ is a partial distance-2 coloring of $G_b(A)$ on V_2 if and only if ϕ induces a structurally orthogonal partition of the columns of A.*

In view of Theorem 3.5, Problem 3.3 is equivalent to the following graph coloring problem.

PROBLEM 3.6. *Given the bipartite graph $G_b(A) = (V_1, V_2, E)$ representing the sparsity structure of an $m \times n$ matrix A, find a partial distance-2 coloring of $G_b(A)$ on V_2 that uses the fewest colors.*

Note that for matrices with a few *dense* rows, a row partition may yield fewer groups than a column partition. Consequently, the matrix problem one needs to solve is Problem 3.3 applied to A^T. In such cases, the bipartite graph formulation becomes handy—the equivalent problem is to find a partial distance-2 coloring in the same graph, but now on the vertex set V_1.

3.3. An Alternative Coloring Formulation. Coleman and Moré [30] were the first to formulate Problem 3.3 as a graph coloring problem. They showed Problem 3.3 to be equivalent to the distance-1 graph coloring problem on the column intersection graph of the underlying matrix. Here we will show the equivalence between the bipartite graph–based distance-2 coloring formulation and the column intersection graph–based distance-1 coloring formulation of Problem 3.3.

The *power* of a graph gives an alternative view to the distance-k graph coloring problem. Recall that the kth power of a graph $G = (V, E)$ is the graph $G^k = (V, F)$ where $(u, v) \in F$ if and only if u and v are distance-k neighbors in G. As stated in Lemma 2.1, a distance-k coloring of G is equivalent to a distance-1 coloring of G^k.

A particular implication of Lemma 2.1 is that a distance-2 coloring of a graph is equivalent to a distance-1 coloring of the square of the graph. This establishes the equivalence between distance-2 coloring the column vertices of the bipartite graph and distance-1 coloring the vertices of the column intersection graph of a matrix. Specifically, the column intersection graph $G_c(A)$ of a matrix A is isomorphic to the adjacency graph of $A^T A$. We note that $G_c(A)$ is in fact the subgraph of ${G_b(A)}^2$ induced by the vertices in V_2.

LEMMA 3.7. *Let $G_b(A) = (V_1, V_2, E)$ and $G_c(A) = (V_2, E')$ be the bipartite and column intersection graphs of matrix A. Then $G_c = {G_b}^2[V_2]$.*

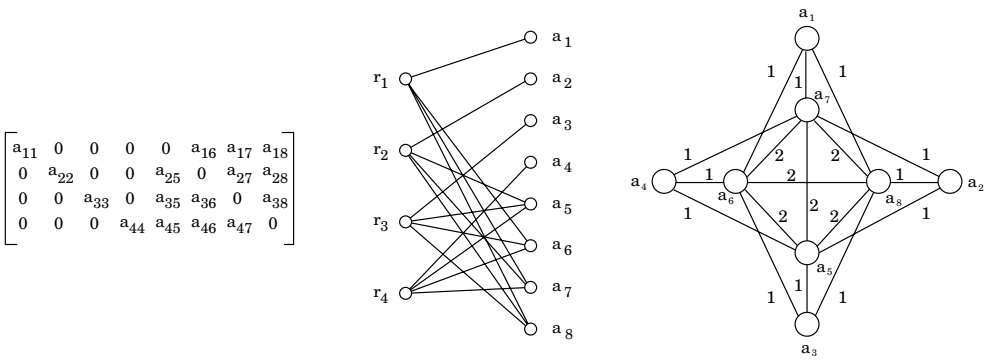

Fig. 3.1 *A matrix and its bipartite and column intersection graph representations.*

3.4. Comparison. Figure 3.1 depicts a matrix A, the corresponding bipartite graph $G_b(A) = (V_1, V_2, E)$, and the column intersection graph $G_c(A) = (V_2, E')$. Note that we have augmented the graph $G_c(A)$ with *edge weights*—$w(a_i, a_j)$ is the size of the intersection of the sets represented by vertices a_i and a_j. In terms of the matrix A, the weight $w(a_i, a_j)$ is the total number of rows where both the ith and the jth columns of A have nonzero entries. In the bipartite graph $G_b(A)$, the weight $w(a_i, a_j)$ corresponds to the number of common neighbors of vertices a_i and a_j.

In sections 3.4.1 through 3.4.5, we compare and contrast the two graph formulations in terms of flexibility, suitability for unification, graph size, ease of graph construction, and use of existing software.

3.4.1. Flexibility. Notice that the column intersection graph is not an equivalent representation of the sparsity structure of the underlying matrix. In particular, given an edge between two column vertices, one cannot determine the row index (indices) at which the columns share nonzero entries. In contrast, the bipartite graph is an equivalent representation of the structure of the matrix. This provides flexibility. For instance, the bipartite graph can be used in a column-only, row-only, or combined row and column partition of the matrix. The column intersection graph, on the other hand, is applicable only to a column partition. In general, the advantage of the bipartite graph representation (in the context of computing derivative matrices) is that the representation is decoupled from the eventual technique to be employed and the matrix entries to be determined.

3.4.2. Unification. As has been discussed in the introduction, depending on the type of matrix to be evaluated and on the method employed, there exist several variants of coloring problems in the context of derivative matrix computation. One of the key advantages of the bipartite graph–based formulation for Jacobian estimation in this regard is that it leads to the identification of an archetypal problem: the distance-2 coloring problem. The identification of an archetype is particularly useful in developing new algorithms and software for the specialized variants.

3.4.3. Graph Size. Although Lemma 3.7 correlates the bipartite graph of a matrix with its column intersection graph, one cannot immediately deduce that one graph is larger in size than the other. The size of the particular graph depends on the sparsity structure of the matrix. Here we provide a rough analysis to show that for sparse matrices of practical interest, the column intersection graph is likely to have many more edges than the bipartite graph representation.

Let A be a matrix, $G_b(A) = (V_1, V_2, E)$ be the bipartite graph of A, and $G_c(A) = (V_2, E')$ be the (weighted) column intersection graph of A. Further, let $\overline{w}$ denote the *average edge weight* in G_c. Then

$$\sum_{e \in E'} w(e) = \frac{1}{2} \cdot \sum_{u \in V_2} \sum_{v \in N_1(u)} (d_1(v) - 1).$$

Expressing the sum on the left-hand side with average weight and bounding $d_1(v)$ by $\Delta(V_1)$, we get

$$|E'| \cdot \overline{w} \leq \frac{1}{2}(\Delta(V_1) - 1) \sum_{u \in V_2} d_1(u).$$

Expressing the sum on the right-hand side with average degree gives

$$|E'| \cdot \overline{w} = O(|V_2| \cdot \overline{\delta}_1(V_2) \cdot \Delta(V_1)).$$

Since $|V_2| \cdot \overline{\delta}_1(V_2) = |E|$, we have

$$|E'| = O(|E| \cdot \Delta(V_1)/\overline{w}). \tag{3.2}$$

Therefore, since the inequality $\Delta(V_1) > \overline{w}$ is likely to hold in practice, and the constant in the Big Oh is close to 1, the column intersection graph of the matrix could have many more edges than the bipartite graph. The larger graph size has in turn two implications. First, more storage space would be required. Second, the average runtime per edge spent in constructing the graph is likely to be larger since more data has to be stored in memory, and memory access time is nonuniform on a hierarchical memory processor.

3.4.4. Ease of Construction. The sparsity structure of the matrix A is identical to the adjacency lists of the vertices of the bipartite graph $G_b(A)$, and hence the construction of the latter is trivial. In principle, the data structure used to represent A could be used for implementing algorithms that use $G_b(A)$. In contrast, $G_c(A)$ has to be computed. The time required for the computation of $G_c(A)$ is proportional to the number of edges in $G_c(A)$. Using (3.2) and noting that $\overline{w} \geq 1$, the following lemma gives the complexity of constructing $G_c(A)$.

LEMMA 3.8. *Given a graph $G_b(A) = (V_1, V_2, E)$, the time required for constructing $G_c(A)$ is $O(|E| \cdot \Delta(V_1))$.*

It should, however, be noted that once the graph $G_c(A)$ is computed, a subsequent distance-1 coloring of $G_c(A)$ can be done faster than a distance-2 coloring of $G_b(A)$. In section 3.5, we show that the overall time complexity of constructing and then distance-1 coloring $G_c(A)$ is of the same order as that of distance-2 coloring $G_b(A)$.

3.4.5. Use of Existing Software. Serial program packages that implement various practically effective distance-1 coloring heuristics exist [28, 29]. For matrix partitioning problems where a column intersection graph–based formulation can be applied, these packages can be readily used. On the other hand, since distance-2 coloring is an archetypal model in our context, efficient programs, including parallel ones, for the distance-2 coloring problem need to be developed.

3.5. Algorithms. For every fixed integer $k \geq 1$, the distance-k graph coloring problem is NP-hard [89]. Thus, in practice, one is bound to rely on approximation algorithms or heuristics. An algorithm $\mathcal{A}$ is said to be a γ-*approximation algorithm* for

ALGORITHM 3.1. A greedy distance-2 coloring algorithm.

```
procedure D2ColoringAlg(G = (V, E))
    Let v_1, v_2, ..., v_|V| be a given ordering of V
    Initialize forbiddenColors with some value a ∉ V
    for i ← 1 to |V| do
        for each colored vertex w ∈ N_1(v_i) do
            forbiddenColors[color[w]] ← v_i
            for each colored vertex x ∈ N_1(w) do
                forbiddenColors[color[x]] ← v_i
            end for
        end for
        color[v_i] ← min{c > 0 : forbiddenColors[c] ≠ v_i}
    end for
end procedure
```

a minimization problem if its runtime is polynomial in the input size and if for every problem instance $\mathcal{I}$ with the value $OPT(\mathcal{I})$ of an optimal solution, the value $\mathcal{A}(\mathcal{I})$ of the solution produced by $\mathcal{A}$ is such that $\frac{\mathcal{A}(\mathcal{I})}{OPT(\mathcal{I})} \leq \gamma$. For a minimization problem, the *approximation ratio* γ satisfies $\gamma \geq 1$, and the goal in designing an approximation algorithm is to make γ as close to unity as possible. If no such guarantee can be given for the quality of an approximate solution obtained by a polynomial time algorithm, the algorithm is usually referred to as a heuristic.

The current best known approximation ratio for the distance-1 coloring problem is $O(|V|(\frac{\log\log|V|}{\log|V|})^3)$ [59]. Also, the problem is known to be not approximable within $O(|V|^{1/7-\epsilon})$ for any $\epsilon > 0$, unless P = NP [12]. Despite these rather pessimistic results, there exist several practically effective distance-1 coloring heuristics [30, 94]. A detailed discussion of some of these heuristics is included in section 11 of this paper. In the current section we show how a distance-1 coloring heuristic can be easily adapted to the distance-2 coloring case by looking at the extended neighborhood of a vertex. The algorithms we present in this paper are *greedy* in nature; i.e., the vertices of a graph are processed in some order and at each step a decision that looks best at the moment (and that will not be changed later) is made.

In section 3.5.1 we present a generic greedy distance-2 coloring algorithm and give a detailed analysis of its performance in terms of both computation time and number of colors used. In later sections, adaptations of this algorithm tailored to the various coloring problems that concern us will be presented.

3.5.1. A Distance-2 Coloring Algorithm. A simple approach for an approximate distance-2 coloring of a graph $G = (V, E)$ is to visit the vertices in some order, at each step assigning a vertex the smallest color that is not used by any of its distance-2 neighbors.

The degree-2 of a vertex v in G is bounded by Δ^2; i.e., $d_2(v) = \sum_{w \in N_1(v)} d_1(w) \leq \Delta \cdot d_1(v) \leq \Delta^2$. The vertices in G can always be distance-2 colored trivially using $|V|$ different colors. Thus, it is always possible to assign a vertex a color chosen from the set $\{1, 2, \ldots, \min\{\Delta^2 + 1, |V|\}\}$. D2ColoringAlg, outlined in Algorithm 3.1, uses this fact as it colors the vertices of the graph in a given order. In the algorithm, color is a vertex-indexed array that stores the color of each vertex and forbiddenColors is a color-indexed array of size $C_{max} = \min\{\Delta^2 + 1, |V|\}$ used to mark the colors

that cannot be assigned to a particular vertex. Specifically, forbiddenColors$[c] = v_i$ indicates that color c cannot be assigned to vertex v_i.

We prove the following statement regarding D2ColoringAlg.

Lemma 3.9. D2ColoringAlg *finds a distance-2 coloring of a graph* $G = (V, E)$ *in time* $O(|V|\bar{\delta}_2)$.

Proof. We first show correctness. In step i of the algorithm, the colors used by each of the colored distance-2 neighbors of vertex v_i are marked (using v_i) in the array forbiddenColors. Thus, at the end of the middle for-loop, the colors that are allowed for vertex v_i are the indices in forbiddenColors, where the marker used is different from v_i. The minimum value in this set is thus the smallest allowable color for vertex v_i. Notice that the array forbiddenColors does not need to be initialized at every step as the marker v_i is used *only* in step i.

Turning to complexity, marking the forbidden colors at step i of the algorithm takes $O(d_2(v_i))$ time. Finding the smallest allowable color to assign v_i can be done within the same order of time by scanning the array forbiddenColors sequentially until the first index c where a value other than v_i is found. The total time is thus proportional to $\sum_{v_i \in V} d_2(v_i) = O(|V|\bar{\delta}_2)$. ∎

We now analyze the quality of the output of D2ColoringAlg. The distance-1 neighbors of a vertex in a graph G form a *clique* in the square graph G^2. This fact immediately provides a lower bound on $\chi_2(G)$, the distance-2 chromatic number of G.

Lemma 3.10. *For every graph* G, $\chi_2(G) \geq \Delta + 1$.

In general, the graph G^2 may contain a clique whose size is larger than $\Delta + 1$. Thus, the *maximum* clique size in G^2 is a tighter lower bound on $\chi_2(G)$ than the bound $\Delta + 1$.

Let the number of colors used by D2ColoringAlg on a graph $G = (V, E)$ be $\chi_2(G, \text{greedy})$. As discussed earlier, $\chi_2(G, \text{greedy}) \leq \min\{\Delta^2 + 1, |V|\}$. Combining this with the lower bound on χ_2 given in Lemma 3.10, we get the following theorem and its corollary. The latter is due to McCormick [97].

Theorem 3.11. $\Delta + 1 \leq \chi_2(G) \leq \chi_2(G, \textit{greedy}) \leq \min\{\Delta^2 + 1, |V|\}$.

Corollary 3.12. D2ColoringAlg *is an* $O(\min\{\Delta, \sqrt{|V|}\}) = O(\sqrt{|V|})$ *approximation algorithm.*

Proof. From Theorem 3.11, the approximation ratio γ is at most $\frac{1}{\Delta+1} \cdot \min\{\Delta^2 + 1, |V|\}$. There are two possibilities to consider. In the first case $\Delta^2 + 1 \leq |V|$. This implies $\Delta = O(\sqrt{|V|})$ and $\gamma \leq \frac{\Delta^2+1}{\Delta+1} = O(\Delta) = O(\sqrt{|V|})$. In the second case $|V| < \Delta^2 + 1$. This implies $\Delta = \Omega(\sqrt{|V|})$ and $\gamma \leq \frac{|V|}{\Delta+1} = O(\sqrt{|V|})$. ∎

For many practical problems, such as meshes from finite element discretization of PDEs, $\Delta^2 \ll |V|$, making D2ColoringAlg an $O(\Delta)$-approximation algorithm.

The actual number of colors used by D2ColoringAlg depends on the order in which the vertices are visited. In our analysis, we assumed an arbitrary ordering. A solution with fewer colors can be expected if a more elaborate ordering criterion is used. For example, orderings such as *largest first*, *smallest last*, *incidence degree*, and *saturation degree*, appropriately adapted, are likely to be as fruitful for distance-2 coloring as for distance-1 coloring. In the context of distance-1 coloring, these orderings are analyzed in section 11.1. For a remark on the adaptation of these orderings for distance-k coloring problems, see section 12.2.

3.5.2. A Partial Distance-2 Coloring Algorithm. D2ColoringAlg needs to be modified only slightly to give an algorithm for the partial distance-2 coloring problem, our graph formulation of Problem 3.3. PartialD2ColoringAlg, given in Algorithm 3.2, is a result of such a modification.

ALGORITHM 3.2. A greedy partial distance-2 coloring algorithm.

```
procedure PartialD2ColoringAlg(G_b = (V_1, V_2, E))
    Let v_1, v_2, ..., v_{|V_2|} be a given ordering of V_2
    Initialize forbiddenColors with some value a ∉ V_2
    for i ← 1 to |V_2| do
        for each w ∈ N_1(v_i) do
            for each colored vertex x ∈ N_1(w) do
                forbiddenColors[color[x]] ← v_i
            end for
        end for
        color[v_i] ← min{c > 0 : forbiddenColors[c] ≠ v_i}
    end for
end procedure
```

Let $G_b = (V_1, V_2, E)$ be an input graph to this algorithm. For every vertex $v \in V_2$, the number of vertices at distance *exactly* two edges from v is at most $\Delta(V_2) \cdot (\Delta(V_1) - 1)$. Thus, vertex v can always be assigned a color from the set $\{1, 2, \ldots, C_{max}\}$, where $C_{max} = \min\{\Delta(V_2) \cdot (\Delta(V_1) - 1) + 1, |V_2|\}$. Hence the array forbiddenColors in PARTIALD2COLORINGALG is of this size.

The time complexity result stated in Lemma 3.13 is evident.

LEMMA 3.13. PARTIALD2COLORINGALG *finds a partial distance-2 coloring of a bipartite graph* $G_b = (V_1, V_2, E)$ *on the vertex set* V_2 *in time* $O(|V_2| \cdot \overline{\delta}_1(V_2) \cdot \Delta(V_1)) = O(|E| \cdot \Delta(V_1))$.

Let $\chi_2(G_b, V_2)$ denote the minimum number of colors required for a partial distance-2 coloring of $G_b = (V_1, V_2, E)$ on V_2. One can then state the following analogous result to Lemma 3.10.

LEMMA 3.14. *For every bipartite graph* $G_b = (V_1, V_2, E)$, $\chi_2(G_b, V_2) \geq \Delta(V_1)$.

In terms of Problem 3.3, Lemma 3.14 says that $\Delta(V_1)$, which is equal to the maximum number of nonzeros in a row of the Jacobian, is a lower bound on the least number of groups required in a structurally orthogonal column partition. This lower bound was observed by Coleman and Moré [30]. It should be noted that the size of a *largest* clique in ${G_b}^2[V_2]$ is a tighter lower bound than $\Delta(V_1)$.

3.5.3. A Distance-1 Coloring Algorithm. As mentioned earlier, Problem 3.3 can also be formulated as a distance-1 coloring problem on the column intersection graph. For comparison purposes, we consider D1COLORINGALG, the distance-1 analog of D2COLORINGALG. The pseudocode for D1COLORINGALG (omitted here) differs from that of D2COLORINGALG in only two ways: (1) the innermost for-loop is skipped; and (2) the array forbiddenColors is of size $\Delta + 1$, instead of $\min\{\Delta^2 + 1, |V|\}$.

The following result is straightforward.

LEMMA 3.15. D1COLORINGALG *finds a distance-1 coloring of a graph* $G = (V, E)$ *in time* $O(|V|\overline{\delta}_1) = O(|E|)$.

From Lemmas 3.8, 3.13, and 3.15 it follows that the time required for constructing and then distance-1 coloring the column intersection graph is asymptotically the same as the time required for distance-2 coloring the column vertices of the bipartite graph. This means that the two formulations are asymptotically comparable in terms of overall computation time. In practice, however, the runtimes while using the two approaches may differ considerably. Our experimental results reported in the next subsection demonstrate this fact.

Table 3.1 *Matrix statistics.*

Matrix	m	n	nnz	κ_{max}	κ_{min}	κ_{avg}	ρ_{max}	ρ_{min}	ρ_{avg}
lp_cre_a	3,516	7,248	18,168	14	1	2.51	360	0	5.17
lp_dfl001	6,071	12,230	35,632	14	1	2.91	228	2	5.87
lp_ken_11	14,694	21,349	49,058	3	1	2.30	122	1	3.34
lp_stocfor3	16,675	23,541	76,473	18	1	3.25	15	1	4.59
lp_ken_13	28,632	42,659	97,246	3	1	2.28	170	1	3.40
lp_pds_10	16,558	49,932	107,605	3	1	2.16	96	1	6.50
lp_maros_r7	3,136	9,408	144,848	46	1	15.4	48	5	46.2
lhr10	10,672	10,672	232,633	36	1	21.8	63	1	21.8
lp_pds_20	33,874	108,175	232,647	3	1	2.15	96	0	6.87
lp_cre_d	8,926	73,948	246,614	13	1	3.33	808	0	27.6
lp_cre_b	9,648	77,137	260,785	14	1	3.38	844	0	27.0
e30r2000	9,661	9,661	306,356	62	8	31.7	62	8	31.7
lhr14	14,270	14,270	307,858	36	1	21.6	63	1	21.6
lp_ken_18	105,127	154,699	358,171	3	1	2.31	325	1	3.40
af23560	23,560	23,560	484,256	21	10	20.6	21	11	20.6
e40r0100	17,281	17,281	553,956	62	8	32.1	62	8	32.1
cage11	39,082	39,082	559,722	31	3	14.3	31	3	14.3
lhr34	35,152	35,152	764,014	36	1	21.7	63	1	21.7
lhr71c	70,354	70,304	1,528,092	36	1	21.7	63	1	21.7
cage12	130,228	130,228	2,032,536	33	5	15.6	33	5	15.6
lp_osa_07	1,118	25,067	144,812	6	1	5.78	17,613	18	130
lp_fit2d	25	10,524	129,042	17	1	12.3	10,500	1,427	5,162

3.6. Experimental Results. Here we report experimental results that compare the bipartite graph–based distance-2 coloring formulation of Problem 3.3 with the column intersection graph–based distance-1 coloring formulation. The comparison focuses on the number of colors, overall execution time, and storage space required by the respective algorithms.

The algorithms in our test (including those reported in section 4.4) were implemented in the programming language C, and the experiments were conducted on an Intel Pentium 4, 2.53 GHz machine with 1 GB memory and 512 KB cache, running Linux 2.4.20/RedHat 8.0.

The vertices are visited in their natural order when greedy coloring algorithms are used in our experiments.

3.6.1. Test Matrices. Our testbed consists of matrices obtained from the University of Florida Sparse Matrix Collection [36]. Table 3.1 lists the test matrices along with some relevant structural statistics. Matrices with the prefix *lp* in their names are linear programming problems translated from Netlib. (For brevity, we drop the prefix *lp* when we refer to these matrices in later tables and figures.) The *lhr*-matrices arise in steady-state chemical process simulation problems. The *cage*-matrices are models used in DNA electrophoresis. Matrices e30r2000 and e40r0100 come from fluid dynamics problems, and matrix af23560 is an airfoil problem.

In Table 3.1, the numbers of rows, columns, and nonzeros in each matrix are listed under m, n, and nnz, respectively. The matrices are sorted in increasing order of number of nonzeros. The maximum, minimum, and average number of nonzeros per *column* of a matrix are listed under κ with an appropriate subscript. The corresponding figures per *row* are given under the various ρ's.

3.6.2. Results and Discussion. Table 3.2 lists results obtained by running PartialD2ColoringAlg on the bipartite graph representation of the test matrices.

Table 3.2 *Results of partial d2 coloring the bipartite graph $G_b(A)$.*

Matrix	$\|V\|$	$\|E\|$	Δ	δ	$\overline{\delta}$	K (ρ_{max})	T_{G_b}	T_{col}	T_{tot}
cre_a	10,764	18,168	360	0	3.38	360 (360)	0	0.01	0.01
dfl001	18,301	35,632	228	1	3.89	228 (228)	0	0.01	0.01
ken_11	36,043	49,058	122	1	2.72	130 (122)	1	0.00	1.00
stocfor3	40,216	76,473	18	1	3.80	16 (15)	1	0.03	1.03
ken_13	71,291	97,246	170	1	2.73	176 (170)	1	0.02	1.02
pds_10	66,490	107,605	96	1	3.23	96 (96)	1	0.01	1.01
maros_r7	12,544	144,848	48	1	23.1	74 (48)	1	0.07	1.07
lhr10	21,344	232,633	63	1	21.8	65 (63)	1	0.10	1.10
pds_20	142,049	232,647	96	0	3.28	96 (96)	2	0.03	2.03
cre_d	82,874	246,614	808	0	5.95	813 (808)	2	0.34	2.34
cre_b	86,785	260,785	844	0	6.01	845 (844)	2	0.34	2.34
e30r2000	19,322	306,356	62	8	31.7	65 (62)	2	0.07	2.07
lhr14	28,540	307,858	63	1	21.6	65 (63)	2	0.11	2.11
ken_18	259,826	358,171	325	1	2.76	330 (325)	7	0.23	7.23
af23560	47,120	484,256	21	10	20.6	32 (21)	3	0.09	3.09
e40r0100	34,562	553,956	62	8	32.1	66 (62)	4	0.13	4.13
cage11	78,164	559,722	31	3	14.3	81 (31)	4	0.12	4.12
lhr34	70,304	764,014	63	1	21.7	65 (63)	6	0.25	6.25
lhr71c	140,608	1,528,092	63	1	21.7	65 (63)	12	0.45	12.4
cage12	260,456	2,032,536	33	5	15.6	96 (33)	15	0.37	15.4
Total1	1,527,603	8,396,670				3,764 (3,573)	67	2.78	69.8
osa_07	26,185	144,812	17,613	1	11.1	17,613 (17,613)	0	5.07	5.07
fit2d	10,549	129,042	10,500	1	24.5	10,501 (10,500)	0	4.63	4.63
Total2	36,734	273,854				28,114 (28,113)	0	9.70	9.70

The left half of the table lists information on the underlying graph. The maximum, minimum, and average degrees are given under Δ, δ, and $\overline{\delta}$, respectively. Note the relationship between the graph and matrix structural statistics given in Tables 3.1 and 3.2: $|V| = m + n$, $|E| = nnz$, $\Delta = \max\{\kappa_{max}, \rho_{max}\}$, and $\delta = \min\{\kappa_{min}, \rho_{min}\}$.

The right half of Table 3.2 lists coloring and timing information. Each test matrix is initially stored as an unordered list of nonzeros (edges). The time for reading this data and for allocating memory for the various graph structures is not included in the reported times. Column K in Table 3.2 lists the number of colors used by PartialD2ColoringAlg, and the number in parenthesis in the same column shows the lower bound ρ_{max} on the optimal number of colors. The time used (in seconds) for constructing the bipartite graph (i.e., building the graph data structure from its equivalent matrix data structure) is given in column T_{G_b}, and the time spent on coloring is listed under T_{col}. The last column gives the sum of the previous two. In Table 3.2, as well as other later tables in this paper, if the time elapsed on some computation is less than 5 ms, we write the value 0.

Table 3.3 lists results obtained when D1ColoringAlg is run on the column intersection graph of each test matrix. The column intersection graph is obtained by first constructing the bipartite graph and then computing the subgraph, induced by the set of column vertices, of the square of the bipartite graph. See Lemma 3.7 for this relationship between the bipartite and column intersection graphs. In our experiments, we used the routine ConstructG_c, outlined in Algorithm 3.3, to compute an intersection graph from a bipartite graph. In ConstructG_c, the array **marked**, which is indexed with vertices in V_2, is used to avoid multiple edges between two vertices in the intersection graph that are connected by several paths of length 2 in the bipartite graph.

Table 3.3 *Results of d1 coloring the column intersection graph $G_c(A)$.*

Matrix	$\|V\|$	$\|E\|$	Δ	δ	$\overline{\delta}$	T_{G_b}	T_{G_c}	T_{col}	T_{tot}
cre_a	7,248	253,411	454	1	69.9	0	0	0.01	0.01
dfl001	12,230	250,976	423	2	41.0	0	4	0.01	4.01
ken_11	21,349	459,921	138	1	43.1	0	2	0.01	2.01
stocfor3	23,541	125,969	39	1	10.7	0	0	0.01	0.01
ken_13	42,659	1,158,664	186	2	54.3	1	9	0.01	10.0
pds_10	49,932	594,681	106	1	23.8	1	8	0.01	9.01
maros_r7	9,408	610,760	314	4	129	1	8	0.01	9.01
lhr10	10,672	431,411	101	1	80.8	1	5	0.02	6.02
pds_20	108,175	1,325,891	115	1	24.5	2	10	0.02	12.0
cre_d	73,948	21,347,885	1,124	2	577	2	76	0.25	78.2
cre_b	77,137	20,852,569	1,168	2	540	2	268	0.75	271
e30r2000	9,661	688,848	209	42	143	2	12	0.02	14.0
lhr14	14,270	572,463	101	1	80.2	1	10	0.00	11.0
ken_18	154,699	8,412,174	340	1	109	8	43	0.15	51.1
af23560	23,560	1,210,004	107	41	103	2	12	0.02	14.0
e40r0100	17,281	1,254,328	209	42	145	3	14	0.01	17.0
cage11	39,082	1,887,384	340	7	96.6	4	16	0.02	20.0
lhr34	35,152	1,417,888	101	1	80.7	5	22	0.03	27.0
lhr71c	70,304	2,835,968	101	1	80.7	8	39	0.04	47.0
cage12	130,228	7,550,823	400	12	116	15	57	0.11	72.1
Total1	930,536	73,242,018				58	615	1.51	675
osa_07	>	100	million	edges					
fit2d	>	100	million	edges					

ALGORITHM 3.3. An algorithm for computing the column intersection graph from a bipartite graph.

```
procedure ConstructG_c(G_b = (V_1, V_2, E))
    E' ← ∅
    for each v ∈ V_2 do
        for each w ∈ N_1(v) do
            for each x ∈ N_1(w) do
                if x ≠ v then
                    if marked[x] ≠ v then
                        E' ← E' ∪ (v, x)
                        marked[x] ← v
                    end if
                end if
            end for
        end for
    end for
    return G_c = (V_2, E')
end procedure
```

In Table 3.3 the time spent on the respective graph constructions is given under columns T_{G_b} and T_{G_c}. The other columns are defined in a similar manner to Table 3.2.

Discussion. A comparison between the results in Tables 3.2 and 3.3 reveals, among others, the following points.

Quality. The two coloring formulations are equivalent in terms of quality. In particular, as long as the vertices are visited in the same order, a greedy distance-1 coloring on a column intersection graph is exactly the same as a partial distance-2 coloring on a bipartite graph. In Table 3.3, the column listing the number of colors used by the algorithm is omitted since the corresponding numbers are identical to those listed under column K in Table 3.2. Another observation regarding the quality of coloring is that, as can be seen from Table 3.2, for the matrices in our experiment, the number of colors used by the greedy algorithm is often close to, and in some cases even equal to, the lower bound ρ_{max} on the optimal value, hence implying optimality.

Graph Size. In general, for the matrices used in our experiments, a column intersection graph is larger in size than a bipartite graph and hence requires more storage space. The difference in graph size becomes particularly large for matrices with relatively "dense" rows ("high" ρ_{avg}). Using the quantity $|V|+|E|$ as a measure of graph size, for instance, the column intersection graph of matrix cre_b is *sixty* times as large as the corresponding bipartite graph. In the case of matrices osa_07 and fit2d, the column intersection graph was too big to fit in the available memory space (the graph construction process in Algorithm 3.3 was stopped when the number of edges $|E'|$ exceeded 100 million). Considering the sum total of the graph sizes of all other matrices from our testbed, the column intersection graph representation would require more than *seven* times as much storage space as the bipartite graph representation. Figure 3.2 shows a comparative plot of graph sizes using the two approaches for a subset of the computational results listed in Tables 3.2 and 3.3.

Time. The average time spent in constructing the bipartite graphs in our experiment is about 8 μs per edge, while the average time spent in constructing the column intersection graphs is about 8.4 μs per edge. The partial distance-2 coloring algorithm traverses on average $|E|\rho_{avg}$ edges, while the distance-1 coloring algorithm traverses $|E|$ edges, where $|E|$ is the number of edges in the appropriate graph in each case. The average time spent on coloring per edge is again roughly the same for the two algorithms with 17 ns for the distance-2 coloring algorithm and 20 ns for the distance-1 coloring algorithm.

In terms of overall computation time for each test case, our experimental results show that a distance-2 coloring approach is significantly faster than a method based on distance-1 coloring (a subset of the results listed in Tables 3.2 and 3.3 is illustrated in Figure 3.3). Again, the difference in overall execution time is large for matrices with dense rows. For example, the overall time used for intersection graph construction and distance-1 coloring for matrix cre_b is 115 times that used for bipartite graph construction followed by partial distance-2 coloring.

For the first twenty matrices, on which both approaches could be tested successfully, the total time spent on constructing and distance-1 coloring the column intersection graph is about *ten* times the overall time spent on constructing and distance-2 coloring the bipartite graph. This rather big difference is most likely due to the fact that a column intersection graph–based approach requires larger memory size, which in turn implies larger memory access time due to the multilevel memory hierarchies in modern microprocessors [39]. A similar explanation can be offered for the observation that building a bipartite graph takes more time than distance-2 coloring it afterwards (see columns T_{G_b} and T_{col} of Table 3.2). It should be pointed out that in our experiments both the bipartite graph– and intersection graph–based approaches rely on "straightforward" implementations; the code was not been optimized by performance-tuning.

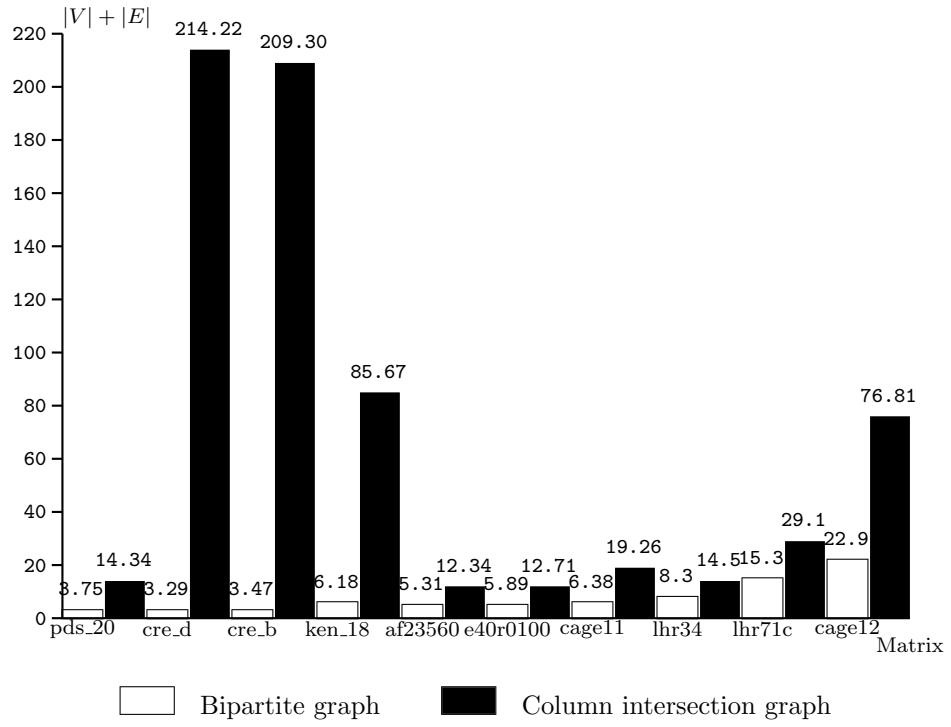

Fig. 3.2 *Graph size comparison. The vertical axis shows* $|V| + |E|$ *scaled by* 100,000.

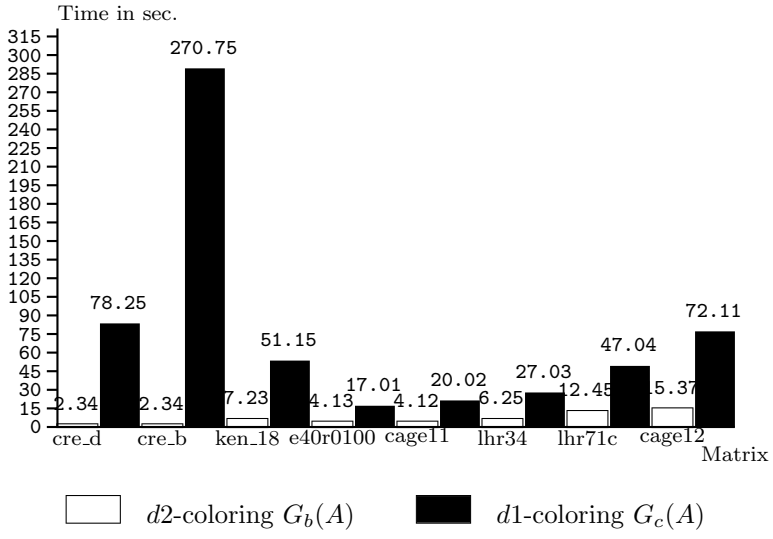

Fig. 3.3 *Overall execution time (graph construction plus coloring) comparison.*

In practice, the time spent on distance-2 coloring a Jacobian is likely to be only a small percentage of the time spent on computing the numerical values of the entries of the Jacobian. For instance, a finite difference evaluation of a Jacobian matrix with a sparsity structure and size similar to that of matrix cage_12 takes about 4 seconds, whereas coloring it takes less than 0.4 seconds [76].

4. Direct Computation of the Hessian. Given a twice continuously differentiable function $f : R^n \to R$, the *Hessian* of f at the point x is the $n \times n$ symmetric matrix whose (i, j) entry $\nabla^2 f(x)_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$. When ∇f is available, $\nabla^2 f$ can be approximated by applying the finite difference formula (3.1) to the function $F = \nabla f$. We assume that evaluating the gradient $\nabla f(x)$ as a single entity is more desirable than evaluating the components $\partial_1 f(x), \ldots, \partial_n f(x)$ separately. In this section, we consider the efficient computation of a Hessian matrix using a direct method.

4.1. The Matrix Partitioning Problem. Let d_i be a linear combination of a subset of the n coordinate vectors $e_j \in R^n$. Analogous to the Jacobian estimation case of the previous section, the problem of interest in the efficient estimation of the Hessian via a direct method can be stated in its general form as follows.

PROBLEM 4.1. *Given the sparsity structure of a symmetric matrix A, find the fewest binary vectors $d_1, d_2, \ldots, d_p$ such that the products $Ad_1, Ad_2, \ldots, Ad_p$ enable a direct determination of A.*

Powell and Toint [107] showed that in the case of Hessian estimation using finite differences, in addition to exploiting sparsity using structurally orthogonal partitions, the number of function evaluations can be reduced further by exploiting symmetry. Coleman and Moré [31] later defined the following more general version of the partition underlying the approach of Powell and Toint.

DEFINITION 4.2. *A partition of the columns of a symmetric matrix A is* symmetrically orthogonal *if for every nonzero element a_{ij}, either* (1) *the group containing the column a_j has no other column with a nonzero in row r_i, or* (2) *the group containing the column a_i has no other column with a nonzero in row r_j.*

A more precise name for the partition referred to in Definition 4.2 would be symmetrically structurally orthogonal; we omit the word *structurally* for the sake of brevity.

The left part of Figure 4.1 illustrates a symmetrically orthogonal partition of the columns of a symmetric matrix. The figure shows a partition of the six columns of the matrix into the three groups $\{a_1, a_3, a_5, a_6\}$, $\{a_2\}$, and $\{a_4\}$.

Let $\{C_1, C_2, \ldots, C_p\}$ be a symmetrically orthogonal partition of the columns of a symmetric matrix A. With each group C_k, associate a binary vector d_k having components $\delta_j = 1$ if a_j belongs to C_k, and $\delta_j = 0$ otherwise. Then

$$Ad_k = \sum_{j=1}^{n} \delta_j a_j = \sum_{a_j \in C_k} a_j.$$

If $a_{ij} \neq 0$ and column a_j is the only column in group C_k with a nonzero in row r_i, then $a_{ij} = (Ad_k)_i$; alternatively, if a_i is the only column in a group $C_{k'}$ with a nonzero in row r_j, then $a_{ji} = (Ad_{k'})_j$. This way, all the diagonal entries and at least one of each pair of symmetric entries of the matrix A can be determined directly with the p evaluations of $Ad_1, \ldots, Ad_p$.

While using a finite difference technique to estimate a symmetric matrix A, if a structurally orthogonal partition rather than a symmetrically orthogonal one is

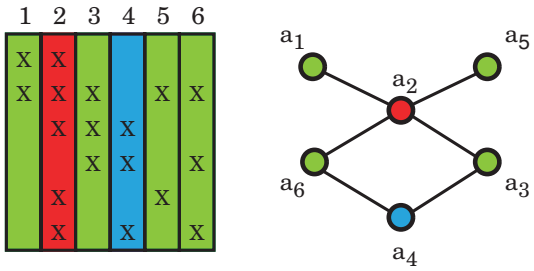

Fig. 4.1 *A symmetrically orthogonal column partition and its representation as a star coloring of the adjacency graph.*

used, the estimate for a_{ij} may differ from that of a_{ji} due to truncation error. Thus, using a symmetrically orthogonal partition to estimate half of the off-diagonal nonzero elements of a matrix and determining the other half by symmetry is preferable in terms of both reducing computational work and ensuring that the computed matrix is indeed symmetric.

Using Definition 4.2, Problem 4.1 can be cast as a partitioning problem as follows.

PROBLEM 4.3. *Given the sparsity structure of a symmetric matrix, find a symmetrically orthogonal partition of its columns that has the fewest groups.*

4.2. A Graph Coloring Formulation. The *adjacency graph* of a symmetric matrix exploits symmetry and hence is an appropriate graph representation for the sparsity structure of a Hessian. A Hessian matrix is often positive definite; hence the assumption in the definition of the associated adjacency graph that all diagonal entries of the matrix are nonzero is reasonable.

We proceed by presenting an adjacency graph–based characterization of structural orthogonality in a symmetric matrix, a result due to McCormick [97].

LEMMA 4.4. *Let A be a symmetric matrix with nonzero diagonal elements, and let $G(A) = (V, E)$ be its adjacency graph. Two columns in A are structurally orthogonal if and only if the corresponding vertices in $G(A)$ are at a distance greater than two from each other.*

Proof. Assume that vertices a_i and a_j are at a distance greater than two from each other in $G(A)$. This implies that (1) $(a_i, a_j) \notin E$, and (2) there exists no path a_i, a_k, a_j in $G(A)$ for any vertex $a_k \in V$, $1 \leq k \leq n$, $k \neq i, j$. In terms of matrix A, this means that (1) $a_{ij} = a_{ji} = 0$, and (2) there is no index $k \in [1, n]$, $k \neq i, j$, such that both a_{ki} and a_{kj} are nonzero. Recall that the diagonal entries a_{ii} and a_{jj} are nonzero. The first statement above ensures that neither row index i nor j causes columns a_i and a_j to be structurally nonorthogonal. The second statement says that there is no other row index k that causes columns a_i and a_j to be nonorthogonal either. Thus, columns a_i and a_j are structurally orthogonal.

Conversely, assume that columns a_i and a_j in A are structurally orthogonal. Noting that $a_{ii} \neq 0$ and $a_{jj} \neq 0$, this implies that (1) $a_{ij} = a_{ji} = 0$, and (2) there is no index k, $1 \leq k \leq n$, $k \neq i, j$, such that $a_{ki} \neq 0$ and $a_{kj} \neq 0$. This means that in $G(A) = (V, E)$, (1) $(a_i, a_j) \notin E$, and (2) there exists no path a_i, a_k, a_j for any vertex $a_k \in V$, $1 \leq k \leq n$, $k \neq i, j$. Hence, vertices a_i and a_j are at a distance $d > 2$ from each other. In particular, they are a distance $d \geq 3$ apart from each other. ∎

Notice that just as structural orthogonality of two columns in a nonsymmetric matrix corresponds to the associated vertices in the bipartite graph being at a dis-

tance greater than two from each other, structural orthogonality of two columns in a symmetric matrix corresponds to the associated vertices in the adjacency graph being at a distance greater than two from each other. There is, however, one difference between the two cases: the distance between a pair of column vertices in a bipartite graph is always even, while the distance between a pair of vertices in an adjacency graph could be odd or even.

By Lemma 4.4, finding a structurally orthogonal partition of the columns of a symmetric matrix A is equivalent to finding a distance-2 coloring of the adjacency graph $G(A)$.

We note that Lemma 4.4 is also applicable to a Jacobian matrix that is symmetric in its zero-nonzero structure, but not in the numerical values of its entries. In particular, distance-2 coloring the adjacency graph of a structurally symmetric Jacobian matrix with nonzero diagonal entries is equivalent to a structurally orthogonal column partition.

In the case of Hessian estimation, the fact that entries a_{ij} and a_{ji} of matrix A are equal in numerical value can be exploited to further reduce the number of groups (colors) required. We now consider the graph coloring formulation of Problem 4.3, the partitioning problem where symmetry is exploited.

Consider a symmetric matrix A with nonzero diagonal elements and let a_{ij}, $i \neq j$, be any nonzero element in A. Further, let a_{ki}, $k \neq i, j$, and a_{jl}, $l \neq i, j, k$, be any other two nonzero elements. These three nonzeros (and their symmetric counterparts) and the relevant diagonal entries of the matrix A have the following structure:

$$\begin{pmatrix} a_{ii} & a_{ij} & a_{ik} & \\ a_{ji} & a_{jj} & & a_{j\ell} \\ a_{ki} & & a_{kk} & \\ & a_{\ell j} & & a_{\ell\ell} \end{pmatrix}.$$

By Definition 4.2, in a symmetrically orthogonal partition of the columns of A,

R1. each of the column pairs (a_i, a_j), (a_i, a_k), and (a_j, a_l) have their respective elements in two different groups; and

R2. either columns a_j and a_k belong to two different groups, or columns a_i and a_l belong to two different groups.

In fact, these two requirements are also sufficient for a symmetrically orthogonal partition. Notice that a partition of the four columns a_i, a_j, a_k, and a_l into the three groups $\{a_i\}$, $\{a_j, a_k\}$, and $\{a_l\}$ satisfies R1 and R2.

Coleman and Moré [31] characterized the requirements R1 and R2 in terms of a coloring in the associated adjacency graph.

DEFINITION 4.5. *A mapping $\phi : V \to \{1, 2, \ldots, p\}$ is a* star coloring *of the graph $G = (V, E)$ if* (1) *ϕ is a distance-*1 *coloring of G, and* (2) *every path on four vertices uses at least three colors.*

Notice that R1 corresponds to the first requirement of Definition 4.5, and R2 corresponds to the second.

The subfigure on the right in Figure 4.1 shows a star coloring representation of the symmetrically orthogonal partition shown on the left. In Figure 4.1, vertices a_1, a_3, a_5, and a_6 are assigned one color; vertex a_2 is assigned a second color; and vertex a_4 is assigned a third color.

Coleman and Moré [31] refer to the coloring given in Definition 4.5 as *path coloring*. Unfortunately, the term *path coloring* is used in the optical networks literature to refer to an assignment of colors to paths in a graph serves as a theoretical model for

routing problems. Examples of works that discuss such applications include [1, 108], and a recent survey is available in [15]. The term star coloring owes its name to the fact that in such a coloring, every subgraph induced by vertices assigned any two colors is a collection of stars. As an illustration, consider the graph shown in Figure 4.1 and the subgraph induced by vertices a_1, a_2, a_3, a_5, and a_6 therein.

The following theorem formalizes the connection between symmetrically orthogonal partitioning and star coloring. The result follows from the discussion that led to the definition of star coloring.

THEOREM 4.6 (Coleman and Moré [31]). *Let A be a symmetric matrix with nonzero diagonal elements, and let $G(A) = (V, E)$ be its adjacency graph representation. A mapping ϕ is a star coloring of $G(A)$ if and only if ϕ induces a symmetrically orthogonal partition of the columns of A.*

By Theorem 4.6, the following problem is equivalent to Problem 4.3.

PROBLEM 4.7. *Given the adjacency graph $G(A) = (V, E)$ representing the sparsity structure of a symmetric matrix A having nonzero diagonal elements, find a star coloring of $G(A)$ that uses the fewest colors.*

4.3. Algorithms. Coleman and Moré [31] showed that the problem of finding a star coloring with the fewest colors is NP-hard even if the graph is bipartite. Here we discuss two heuristic algorithms for this problem. The first algorithm is new, while the second was proposed earlier by Powell and Toint [107] in terms of matrices. In finding a valid color to assign a vertex, the first algorithm visits the distance-3 neighbors of the vertex while the second algorithm visits only the distance-2 neighbors. The latter essentially solves a more restricted coloring problem. The two algorithms represent a trade-off between the number of colors used and runtime. In both algorithms an array **forbiddenColors** of size $C_{max} = \min\{\Delta^2 + 1, |V|\}$ is used.

While this paper was in print, we developed a new star coloring algorithm whose time complexity is the same as the second algorithm discussed here and whose performance is similar to the first. See section 12.2 for a remark.

4.3.1. The First Star Coloring Algorithm. STARCOLORINGALG1 (Algorithm 4.1) outlines the first algorithm. Figure 4.2 graphically shows the decision made during one of the $|V|$ steps of the for-loop in lines 4–21 of STARCOLORINGALG1. In Figure 4.2, the root of the tree corresponds to the vertex v to be colored at the current step. The neighbors of v that are one, two, and three edges away are represented by the tree nodes at levels w, x, and y, respectively. A green-painted node signifies that the vertex is already colored. The forbidden colors are marked by an f and "?" indicates that whether the color is forbidden or not depends on the color used at level y. The correspondence between Figure 4.2 and lines 6, 10, and 14 of Algorithm 4.1 is obvious. Note that each tree node corresponds to many vertices of the input graph.

Line 6 in Algorithm 4.1 guarantees that the resulting coloring is consistent with a distance-1 coloring. Notice that in line 10 of the algorithm, the color of vertex x in a path v, w, x where w is not yet colored is forbidden for vertex v. Later on, when vertex w is colored, the test in line 6 ensures that w will not have two neighbors of the same color and thus will not be the center of a 2-colored path of length two (which could stretch into a 2-colored path of length three). Thus it remains to ensure that v does not become an endpoint of a 2-colored path of length three, and this is done in line 14 of the algorithm. The **break** statement in Algorithm 4.1 is used since the discovery of the first (out of possibly several) 2-colored paths of length two connected to v suffices to force v to have a third color. (A **break** statement forces control to jump out of the nearest loop that contains the statement, in this case, out of the for-loop in lines 13–18.)

ALGORITHM 4.1. A greedy star coloring algorithm.

```
1: procedure StarColoringAlg1(G = (V, E))
2:     Let v_1, v_2, ..., v_|V| be a given ordering of V
3:     Initialize forbiddenColors with some value a ∉ V
4:     for i ← 1 to |V| do
5:         for each w ∈ N_1(v_i) do
6:             if w is colored then
7:                 forbiddenColors[color[w]] ← v_i
8:             end if
9:             for each colored vertex x ∈ N_1(w) do
10:                if w is not colored then
11:                    forbiddenColors[color[x]] ← v_i
12:                else
13:                    for each colored vertex y ∈ N_1(x), y ≠ w do
14:                        if color[y] = color[w] then
15:                            forbiddenColors[color[x]] ← v_i
16:                            break
17:                        end if
18:                    end for
19:                end if
20:            end for
21:        end for
22:        color[v_i] ← min{c > 0 : forbiddenColors[c] ≠ v_i}
23:    end for
24: end procedure
```

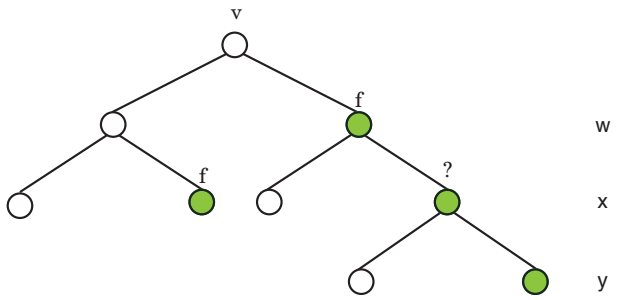

Fig. 4.2 *Visualizing a step in* StarColoringAlg1.

The work done to assign a color to vertex v_i in StarColoringAlg1 is proportional to $d_3(v_i)$. Thus we get the following result.

LEMMA 4.8. StarColoringAlg1 *finds a star coloring of a graph* $G = (V, E)$ *in time* $O(|V|\overline{\delta}_3)$.

4.3.2. The Second Star Coloring Algorithm. Recall that a coloring of a graph is an assignment of positive integers to its vertices. The second star coloring algorithm is based on the observation that a star coloring is a relaxed distance-2 coloring, and makes use of the fact that colors are positive integers.

One way of relaxing the requirement for a distance-2 coloring so as to obtain a star coloring is to let two vertices at distance of exactly two edges from each other

ALGORITHM 4.2. A second greedy star coloring algorithm.

```
1: procedure StarColoringAlg2(G = (V, E))
2:     Let v_1, v_2, ..., v_|V| be an ordering of V
3:     Initialize forbiddenColors with some value a ∉ V
4:     for i ← 1 to |V| do
5:         for each w ∈ N_1(v_i) do
6:             if w is colored then
7:                 forbiddenColors[color[w]] ← v_i
8:             end if
9:             for each colored vertex x ∈ N_1(w) do
10:                if w is not colored then
11:                    forbiddenColors[color[x]] ← v_i
12:                else
13:                    if color[x] < color[w] then
14:                        forbiddenColors[color[x]] ← v_i
15:                    end if
16:                end if
17:            end for
18:        end for
19:        color[v_i] ← min{c > 0 : forbiddenColors[c] ≠ v_i}
20:    end for
21: end procedure
```

share a color as long as the vertex in between them has a color of lower value. More precisely, let v, w, x be a path in G and suppose v and w are colored and we want to determine the color of x. Clearly, we need to make sure that $\phi(x)$ is distinct from $\phi(w)$. Further, we allow $\phi(x)$ to be equal to $\phi(v)$ as long as $\phi(w) < \phi(v)$. To see that this coloring can always be extended to yield a valid star coloring, consider the path v, w, x, y, an extension of the path v, w, x in one direction. Now, since $\phi(x) = \phi(v) > \phi(w)$, we cannot let $\phi(y)$ be equal to $\phi(w)$. Obviously, $\phi(y)$ should be different from $\phi(x)$, otherwise it will not be a valid distance-1 coloring. Thus the path v, w, x, y uses three colors, ϕ is a distance-1 coloring, and therefore it is a valid star coloring. StarColoringAlg2, the algorithm that uses this idea, is outlined in Algorithm 4.2. The work done to assign a color to vertex v_i in StarColoringAlg2 is proportional to $d_2(v_i)$, hence the following result.

LEMMA 4.9. StarColoringAlg2 *finds a star coloring of a graph* $G = (V, E)$ *in time* $O(|V|\overline{\delta}_2)$.

Notice that the coloring produced by StarColoringAlg2 is a more restricted variant of star coloring, and therefore StarColoringAlg1 is likely to use fewer colors than StarColoringAlg2 for a given input graph. Figure 4.3 shows an example where the first algorithm uses three colors in coloring the vertices in their alphabetical order while the second uses four in doing the same. The two star coloring algorithms represent a trade-off between the number of colors used and the computational time needed.

4.4. Experimental Results. Our experimental work in the Hessian estimation case has two objectives: (i) to experimentally demonstrate the advantage of exploiting symmetry, and (ii) to compare and contrast the performance of the two star coloring algorithms.

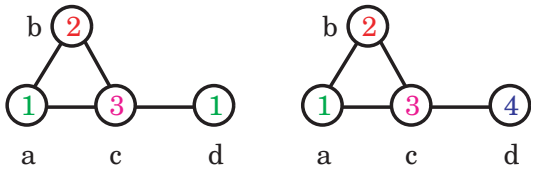

Fig. 4.3 *A simple example where* StarColoringAlg1 *uses fewer colors than* StarColoringAlg2.

Table 4.1 *Graph statistics.*

Graph	$\|V\|$	$\|E\|$	Δ	δ	$\overline{\delta}$
mrng1	257,000	505,048	4	2	3
mrng2	1,017,253	2,015,714	4	2	3
598a	110,971	741,934	26	5	13
144	144,649	1,074,393	26	4	14
m14b	214,765	1,679,018	40	4	15
auto	448,695	3,314,611	37	4	14

Table 4.2 *The number of colors used by and the runtimes of* D2ColoringAlg, StarColoring Alg1, *and* StarColoringAlg2.

Graph	$K(d2)$	$K(star1)$	$K(star2)$	$T(d2)$	$T(star1)$	$T(star2)$
mrng1	12	8	10	0.37	0.61	0.35
mrng2	12	9	10	1.74	2.90	1.75
598a	38	27	32	0.74	3.00	0.75
144	41	28	35	0.90	4.35	0.93
m14b	42	29	34	1.19	5.65	1.23
auto	42	29	36	3.97	17.4	4.07
Total	187	130	157	8.91	33.9	9.08

4.4.1. Test Graphs. The matrices behind the test graphs used in our experiments arise from finite element methods [46]. Table 4.1 gives some structural information about the adjacency graphs of these matrices.

4.4.2. Results and Discussion. Table 4.2 shows the performance of algorithms D2ColoringAlg, StarColoringAlg1, and StarColoringAlg2 when applied to our test graphs. In each algorithm, the vertices are visited in their natural order. The left half of the table shows the number of colors used by the different algorithms and the right half shows the corresponding time (in seconds) spent on coloring.

The results clearly demonstrate the advantage of exploiting symmetry: star coloring requires significantly fewer colors than distance-2 coloring. The total number of colors required by StarColoringAlg1 over all involved test graphs is about 30% less than the number required by D2ColoringAlg. The table also shows nicely the time/quality trade-off between the two star coloring algorithms. StarColoringAlg2 uses nearly the same time as D2ColoringAlg, while the number of colors required is 16% less.

5. Bidirectional, Direct Computation of the Jacobian. Recall that a *bidirectional* partition, as opposed to a *unidirectional* partition, involves both the rows and columns of a matrix. We begin this section by motivating the need for a bidirectional

partition within the context of AD. We then introduce the partitioning problem, discuss its graph coloring formulation, and suggest algorithms for solving the problem.

5.1. The Need for a Bidirectional Partition. Through its forward and reverse mode AD allows a columnwise and a rowwise computation of a Jacobian, respectively. One way of exploiting sparsity in a matrix in this context is to *separately* partition the columns and rows of the matrix, choose the partition which gives the minimum number of groups, and apply the appropriate mode of AD to compute the matrix entries. For a symmetric matrix, a row partition is equivalent to a column partition, but for a nonsymmetric matrix, the two partitions may differ considerably. For example, consider an $n \times n$ matrix where all the entries on the diagonal and the first row are nonzero, and the rest of the matrix entries are all zero. In such a case, a structurally orthogonal column partition requires n groups, whereas a structurally orthogonal row partition requires just two groups.

However, an approach based on a separate row and column partition is not always satisfactory. For example, consider an $n \times n$ arrowhead matrix where all of the elements in the first row, first column, and the diagonal are nonzero and the rest of the entries are all zero. For such a pattern, a structurally orthogonal row partition requires n groups and so does a column partition. However, using a *combined* row and column partition, three groups are enough to determine all the nonzero entries of the matrix. First, separately evaluate the entries in the first column and the first row (two groups). Then, since the remaining $(n-1) \times (n-1)$ matrix is diagonal, group the columns together and determine all entries by one forward AD pass. Thus, three groups (two column and one row) suffice to determine all the nonzero entries.

In the current section we consider such a computation of a nonsymmetric matrix using the combined modes of AD via a direct method. The corresponding problem using a substitution method will be discussed in section 6.2.

Note that a bidirectional partition does not make sense for computing a numerically symmetric matrix. In particular, a symmetry-exploiting unidirectional partition, as discussed in section 4, is sufficient.

5.2. The Matrix Partitioning Problem. Let A be an $m \times n$ matrix with a known sparsity structure. Partitioning the columns of A into p groups can be seen as the task of seeking an $n \times p$ binary matrix D whose (j,k) entry is defined as follows:

$$d_{jk} = \begin{cases} 1 & \text{if column } a_j \text{ belongs to group } k, \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

Recall that D has been referred to as a *seed* matrix in earlier sections. Using this, an alternative way of posing Problem 3.1 would then be the following: Given the structure of an $m \times n$ matrix A, find an $n \times p$ seed matrix D with the least value of p such that the product AD enables a direct determination of A. By the same token, the problem that arises in the bidirectional computation of a Jacobian via a direct method can be posed as follows.

PROBLEM 5.1. *Given the sparsity structure of an $m \times n$ matrix A, find an $n \times p_1$ binary matrix D_1 and an $m \times p_2$ binary matrix D_2 such that the products AD_1 and $D_2^T A$ together enable a direct determination of A and the value $p = p_1 + p_2$ is minimized.*

In this problem formulation, the product AD_1 corresponds to the columns of A determined using the forward mode of AD, and the product $D_2^T A$ corresponds to the rows of A determined using the reverse mode of AD.

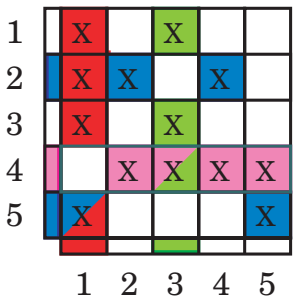

Fig. 5.1 *A structurally orthogonal bidirectional partition of a matrix. The color(s) used on each nonzero entry shows the column and/or row group from which it can be computed.*

Hossain and Steihaug [62] studied Problem 5.1 and identified an associated partitioning problem. Their formulation relies on the notion of a *consistent row-column partition* in which the *entire* set of rows and columns is partitioned into two respective sets of groups. Coleman and Verma [32] also studied the same problem and identified a similar bidirectional partitioning problem. Their notion of partition differs from that of Hossain and Steihaug in that it partitions only a *subset* of the columns and the rows of the matrix that suffices for the direct determination of the entries. We consider such a partition and introduce its formal definition below.

DEFINITION 5.2. *A* bidirectional partition *of a matrix A is a pair (Π_C, Π_R) where Π_C is a partition of a subset of the columns of A and Π_R is a partition of a subset of the rows of A.*

DEFINITION 5.3. *A bidirectional partition (Π_C, Π_R) of a matrix A is* structurally orthogonal *if for every nonzero element a_{ij}, either* (1) *column a_j is in a group of Π_C which has no other column having a nonzero in row r_i, or* (2) *row r_i is in a group of Π_R which has no other row having a nonzero in column a_j.*

The number of column and row groups in a structurally orthogonal bidirectional partition corresponds to the number of forward and reverse AD passes, respectively, required to compute the nonzero entries directly. To see this, observe that a nonzero element a_{ij} can be determined either from a column group where column a_j is the only column with a nonzero in row r_i, or from a row group where row r_i is the only row with a nonzero in column a_j.

Figure 5.1 shows an example of a structurally orthogonal bidirectional partition of a matrix. In the example, Π_C includes columns a_1 and a_3, whereas Π_R includes rows r_2, r_4, and r_5. As can be seen from the color used at the left and bottom edges of the figure, column a_1 forms one column group and column a_3 forms another; similarly, rows r_2 and r_5 form one group and row r_4 forms a second row group. Thus, the bidirectional partition uses a total of four groups. Notice that some entries of the matrix (painted with one color) can be computed only from one group while others (painted with two colors) can be computed from either of the two groups. As the reader can easily verify, for this example, a row-only or a column-only structurally orthogonal partition would have required five groups.

Assuming that the computational costs involved in the forward and reverse modes of AD are of the same order, in an efficient method that uses a bidirectional partition (Π_C, Π_R), the value $|\Pi_C| + |\Pi_R|$ is required to be as small as possible. Thus Problem 5.1 can be cast as a partitioning problem in the following way.

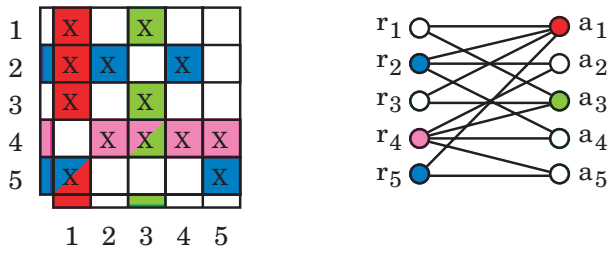

Fig. 5.2 *A structurally orthogonal bidirectional partition and its representation as a star bicoloring. The illustration shows a bidirectional partition* (Π_R, Π_C)*:* $\Pi_R = \{r_2, r_5\}; \{r_4\}$ *and* $\Pi_C = \{a_1\}; \{a_3\}$.

PROBLEM 5.4. *Given the sparsity structure of an* $m \times n$ *matrix* A*, find a structurally orthogonal bidirectional partition* (Π_C, Π_R) *such that* $|\Pi_C| + |\Pi_R|$ *is minimized.*

It should be pointed out that in formulating Problem 5.4 within the context of AD, we are concerned only with *computational* cost. However, in general, the forward mode of AD requires less memory space than the reverse mode, making the former perhaps more desirable. Hence, a more accurate objective would be to minimize $w_1|\Pi_C| + w_2|\Pi_R|$ for some empirically determined *weights* w_1 and w_2 [66].

5.3. A Graph Coloring Formulation. When a nonsymmetric matrix A is represented by its bipartite graph $G_b(A) = (V_1, V_2, E)$, we have seen that a structurally orthogonal unidirectional partition can be obtained by finding a partial distance-2 coloring of G_b on the set V_2 of column vertices. We now consider how this coloring needs to be modified to find a structurally orthogonal bidirectional partition. Notice that the coloring we are looking for has to meet the following conditions.

- Some vertices may not be involved in the determination of any nonzero entry of the underlying matrix. Such vertices are assigned the "neutral" color zero. We use positive integers to denote the other colors.
- The colors assigned to vertices in V_1 should be disjoint from colors assigned to vertices in V_2, except for the neutral color zero.
- Since every nonzero matrix entry has to be determined, for every edge in E, at least one of the endpoints has to be assigned a positive color.
- A nonzero matrix entry may be determined from either a positively colored column vertex or a positively colored row vertex. This suggests that the coloring condition sought here is some relaxation of the distance-2 coloring requirement imposed in the case of unidirectional partition.

The following definition, introduced by Coleman and Verma [32], makes the aforementioned conditions more precise. The subsequent theorem establishes the equivalence between the matrix and graph problems.

DEFINITION 5.5. *Let* $G_b = (V_1, V_2, E)$ *be a bipartite graph. A mapping* $\phi : [V_1, V_2] \to \{0, 1, \ldots, p\}$ *is a* star bicoloring *of* G_b *if the following conditions hold:*

1. *If* $u \in V_1$ *and* $v \in V_2$*, then* $\phi(u) \neq \phi(v)$ *or* $\phi(u) = \phi(v) = 0$.
2. *If* $(u, v) \in E$*, then* $\phi(u) \neq 0$ *or* $\phi(v) \neq 0$.
3. *If vertices* u *and* v *are adjacent to a vertex* w *with* $\phi(w) = 0$*, then* $\phi(u) \neq \phi(v)$.
4. *Every path on four vertices uses at least three colors.*

Figure 5.2 shows a structurally orthogonal bidirectional partition of a matrix and its representation as a star bicoloring in the associated bipartite graph. The reader

ALGORITHM 5.1. A scheme for star bicoloring.

procedure STARBICOLORINGSCHEME($G_b = (V_1, V_2, E)$)
1. Find a suitable vertex cover C in G_b
2. Assign the vertices in the set $I = (V_1 \cup V_2) \setminus C$ the color 0
3. Color the vertices in C such that the result is a star bicoloring of G_b

end procedure

could verify that these four conditions listed above are satisfied by the coloring in the graph.

THEOREM 5.6 (Coleman and Verma [32]). *Let A be an $m \times n$ matrix and $G_b(A) = (V_1, V_2, E)$ be its bipartite graph. The mapping $\phi : [V_1, V_2] \rightarrow \{0, 1, \ldots, p\}$ is a star bicoloring of $G_b(A)$ if and only if ϕ induces a structurally orthogonal bidirectional partition (Π_C, Π_R) of A.*

Thus Problem 5.4 is equivalent to the following graph coloring problem.

PROBLEM 5.7. *Given the bipartite graph $G_b(A) = (V_1, V_2, E)$ representing the sparsity structure of an $m \times n$ matrix A, find a star bicoloring of $G_b(A)$ that uses the fewest colors.*

5.4. Algorithms. In a star bicoloring of a bipartite graph, some vertices are assigned the neutral color zero. (As an example, see the unpainted vertices in Figure 5.2.) We make the following observation which helps us identify a possible set of such vertices. The observation is a direct consequence of conditions 1 and 2 of Definition 5.5.

OBSERVATION 5.8. *Let $G_b = (V_1, V_2, E)$ be a bipartite graph and $\phi : [V_1, V_2] \rightarrow \{0, 1, \ldots, p\}$ be a star bicoloring of G_b. Then*

- *the set $C = \{v : \phi(v) \neq 0\}$ is a* vertex cover *in G_b, and*
- *the set $I = \{v : \phi(v) = 0\}$ is an* independent set *in G_b.*

One consequence of Observation 5.8 is that $|I|+|C| = |V_1|+|V_2|$. Thus, a decrease in the cardinality of the vertex cover C results in an increase in the cardinality of the independent set I.

Observation 5.8 suggests a scheme for solving Problem 5.7. This scheme, called STARBICOLORINGSCHEME, is outlined in Algorithm 5.1.

In step 1 of STARBICOLORINGSCHEME, a vertex cover needs to be chosen carefully. Though a vertex cover of small size is desirable, minimizing its cardinality is not a primary objective. Specifically, the chosen vertex cover should be such that it results in fewer colors compared to both a partial distance-2 coloring on V_1 (rowwise unidirectional partition) and a partial distance-2 coloring on V_2 (columnwise unidirectional partition). As the discussion in section 5.1 suggests, such a vertex cover needs to include vertices from V_1 and V_2 having a relatively high number of distance-1 neighbors, even when the implication is that the vertex cover is not minimal in size. Coleman and Verma [32] have suggested a procedure (formulated in matrix terms) for computing a vertex cover suitable for star bicoloring.

A high-level translation of the procedure of Coleman and Verma in graph terms would be as follows. A vertex cover $C = C_1 \cup C_2$ is obtained by first finding an independent set $I = I_1 \cup I_2$ and then applying the differences $C_1 = V_1 \setminus I_1$ and $C_2 = V_2 \setminus I_2$. The independent set is obtained using an algorithm that has the flavor of a minimum degree algorithm for finding a maximal independent set. An additional ingredient of the algorithm is a classification of the edge set E into four categories. The first category corresponds to edges not yet covered. An edge in the

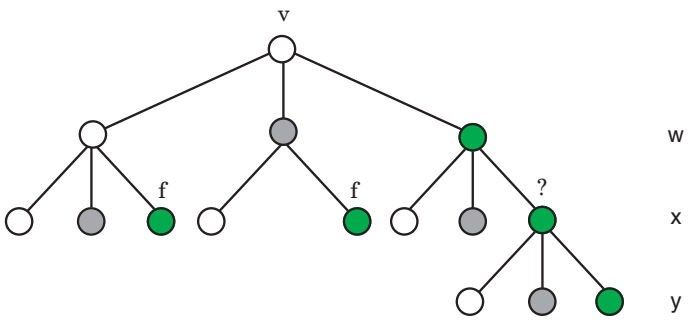

Fig. 5.3 *Visualizing a step in* StarBicoloringAlg.

second category is an edge covered solely by a vertex from V_1, an edge in the third category is covered solely by a vertex from V_2, and an edge in the fourth category is covered both by a vertex from V_1 and by a vertex from V_2. This classification is used to define specialized vertex "degrees" (number of incident edges of each category). Initially all edges are set to be in category 1, to reflect that they are not covered yet. In each step of the greedy algorithm, a vertex v_1 of minimum degree in V_1 and a vertex v_2 of minimum degree in V_2 are chosen; the minimum degree in each case is evaluated using the subgraph of G_b obtained by deleting category-2 and category-3 edges. Then, based on a test function that involves a comparison between current maximum degrees (restricted to category-2 and category-3 edges), one of the vertices v_1 or v_2 is chosen to be included in the independent set. Once the vertex to be included in the independent set is determined, the edges incident on the chosen vertex and its neighbors are placed into appropriate categories that reflect the choice. The algorithm terminates when no category-1 edge is left (all edges are covered).

Once steps 1 and 2 are carried out, step 3 can be done by a suitable adaptation of StarColoringAlg1. Let StarBicoloringAlg be such an adaptation. One of the differences between the coloring and bicoloring algorithms is that in the latter case, two disjoint sets of colors are used in coloring the vertices in V_1 and V_2 of the bipartite graph $G_b = (V_1, V_2, E)$. Another difference is that at the instance of the bicoloring algorithm in which vertex v is colored, a vertex within the distance-3 neighborhood of v may be one of *three* types: it is colored with a positive value, it is colored with 0, or it is not yet colored. The choice of color for vertex v thus needs to consider these three options.

Figure 5.3 shows a visual presentation of an iteration (in the loop over all vertices) of StarBicoloringAlg. Note the similarity with Figure 4.2. Since the colors for the vertices in V_1 and V_2 are chosen from two disjoint sets, in choosing a color for vertex v in $G_b = (V_1, V_2, E)$, we need only consider colors of vertices that are exactly two edges away from v. In the figure, green-painted nodes correspond to vertices with positive colors, shaded nodes show vertices with color zero, and unpainted nodes correspond to uncolored vertices. Observe that the node with color zero at level w has only two children; it cannot have a child with color zero, for otherwise there would exist an uncovered edge. The colors of the vertices in the nodes marked by an f indicate forbidden colors; whether the color at the node marked by "?" is forbidden or not depends on the color used at node y: if $\phi(w) = \phi(y)$, $\phi(x)$ is forbidden; otherwise, it is not.

The time complexity of StarBicoloringAlg is $O((|V_1| + |V_2|)\bar{\delta}_3)$, which is also the overall time complexity of StarBicoloringScheme assuming that step 1 is done using a greedy algorithm that is linear in the number of edges.

Notice that a partial distance-2 coloring of G_b on V_2 is just a special case of StarBicoloringScheme. To see this, consider the trivial choice of vertex cover $C = V_2$ in step 1. This implies that, in step 2, the vertices in the set $I = V_1$ will be colored with color zero. By condition 3 of Definition 5.5, vertices adjacent to a vertex colored with color zero are required to be assigned different colors. Thus, the result is effectively a partial distance-2 coloring of G_b on V_2.

Hossain and Steihaug [62] and Coleman and Verma [32] each proposed an algorithm for Problem 5.7. These can be interpreted in light of StarBicoloringScheme. The algorithm of Hossain and Steihaug *implicitly* finds a vertex cover while the coloring of the graph proceeds. Using our terminology, the vertices that remain uncolored at the end of the Hossain–Steihaug algorithm form an independent set in the graph and can thus be assigned the neutral color zero.

The algorithm of Coleman and Verma uses a preprocessing step to identify the rows and columns of the underlying matrix that eventually need to be colored with positive values. The preprocessing step uses a matrix-based procedure which effectively produces a small-sized vertex cover. After the preprocessing step, certain "column and row intersection graphs," adapted to the star bicoloring requirements, are constructed to finally use known distance-1 coloring heuristics on the resulting graphs.

6. Substitution Methods. In a unidirectional computation of a matrix A via a substitution method, the vectors $d_1, d_2, \ldots, d_p$ are chosen such that the system of equations defined by the products $Ad_1, Ad_2, \ldots, Ad_p$ can be made triangular. In this case, the partition defined by the p vectors could fulfill a more relaxed set of requirements compared to a structurally orthogonal partition used in a direct method, and hence results in a smaller p. This fact has been especially useful when estimating a Hessian matrix since substitution can be effectively combined with the exploitation of symmetry [27, 31]. In the Jacobian case, a substitution method is worth considering when the computation is bidirectional [32].

In this section, within the context of substitution methods, we consider unidirectional partitions for Hessians (section 6.1) and bidirectional partitions for Jacobians (section 6.2).

An example of a variant of a substitution method for a unidirectional computation of a Jacobian has been considered by Hossain and Steihaug [63]. The method relies on first finding a structurally orthogonal partition of the columns and then merging a pair of consecutive groups to get a column grouping that allows overlaps. In this way a structurally orthogonal partition consisting of p groups is changed into a column grouping having $p - 1$ groups. The former defines a diagonal system of equations, while the latter leads to a triangular system.

6.1. Computing the Hessian. To illustrate the fact that a partition used in a substitution method requires fewer groups than one used in a direct method, consider the 4×4 symmetric matrix A shown below:

$$\begin{bmatrix} a_{11} & a_{12} & & \\ a_{21} & a_{22} & a_{23} & \\ & a_{32} & a_{33} & a_{34} \\ & & a_{43} & a_{44} \end{bmatrix}.$$

Any symmetrically orthogonal partition of the columns of this matrix—and hence a direct method—requires at least three groups. An example of a symmetrically orthogonal partition is $\{a_1, a_3\}$, $\{a_2\}$, and $\{a_4\}$. However, if we do not insist on

determining the elements directly, two groups would suffice. For example, consider the partition $\{a_1, a_3\}$ and $\{a_2, a_4\}$. The seed matrix corresponding to this partition is

$$D^T = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

The product AD from which the nonzeros of A can be recovered is

$$AD = \begin{bmatrix} a_{11} & a_{12} \\ a_{12} + a_{23} & a_{22} \\ a_{33} & a_{23} + a_{34} \\ a_{34} & a_{44} \end{bmatrix}.$$

Thus, nonzero entries a_{11}, a_{12}, a_{22}, a_{33}, a_{34}, and a_{44} can be obtained directly and element a_{23} can be obtained via substitution either from the expression $a_{12} + a_{23}$ or from $a_{23} + a_{34}$. Note that in the product AD, every nonzero entry a_{ij} has been identified with its symmetric counterpart a_{ji}.

6.1.1. The Matrix Partitioning Problem. In general, a column partition of a symmetric matrix induces a substitution method if there is an ordering of the nonzero entries (unknowns) such that all unknowns can be solved for, in that order, using symmetry and previously solved elements.

We formally define such a partition and then state the corresponding partitioning problem. In the following definition, we identify the two matrix elements a_{ij} and a_{ji} of the symmetric matrix A. The ordering of the elements in the definition is the ordering in which the matrix elements are evaluated in a substitution method.

DEFINITION 6.1. *A partition of the columns of a symmetric matrix A is said to be* substitutable *if there exists an ordering on the elements of A such that for every nonzero a_{ij}, either* (1) *column a_j is in a group where all the nonzeros in row r_i, from other columns in the same group, are ordered before a_{ij}, or* (2) *column a_i is in a group where all the nonzeros in row r_j, from other columns in the same group, are ordered before a_{ij}.*

PROBLEM 6.2. *Given the sparsity structure of a symmetric matrix, find a substitutable partition of its columns that has the fewest groups.*

6.1.2. A Graph Coloring Formulation. To formulate Problem 6.2 as a graph problem, we need to introduce the notion of *acyclic* coloring.

DEFINITION 6.3. *A mapping $\phi : V \to \{1, 2, \ldots, p\}$ is an acyclic coloring of a graph $G = (V, E)$ if* (1) *ϕ is a distance-*1 *coloring, and* (2) *every cycle in G uses at least three colors.*

Note that since the vertices in every cycle are assigned at least three colors in an acyclic coloring, the subgraph induced by the set of vertices assigned any two colors does not contain a cycle, and hence it is a forest. The name acyclic coloring stems from this fact.

Consider the 10×10 symmetric matrix A whose nonzero structure is depicted in Figure 6.1. The figure shows a partition of the columns of A into the three groups $\{a_1, a_3, a_5\}$, $\{a_2, a_4, a_7, a_9\}$, and $\{a_6, a_8, a_{10}\}$. We will show that this partition is substitutable.

An acyclic coloring of the adjacency graph $G(A)$ is shown in Figure 6.1 (right); the vertices $\{a_1, a_3, a_5\}$ are colored red, the vertices $\{a_2, a_4, a_7, a_9\}$ are colored blue, and the vertices $\{a_6, a_8, a_{10}\}$ are colored green. Note that the subgraph induced

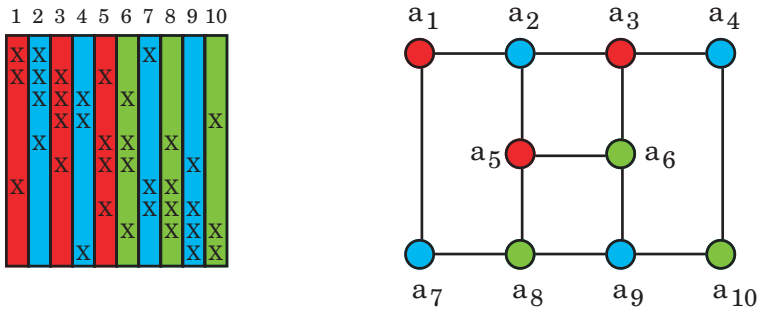

Fig. 6.1 *A substitutable partition of the columns of a symmetric matrix and its representation as an acyclic coloring of the associated adjacency graph.*

by the vertices colored red or blue, the set $\{a_1, a_2, a_3, a_4, a_5, a_7, a_9\}$, is a forest; the vertex a_9 is an isolated vertex in this forest and the remaining six vertices constitute a tree. Similarly, the subgraph induced by the vertices colored blue or green is a forest consisting of the tree on the six vertices $a_4, a_6, a_7, a_8, a_9, a_{10}$ and the isolated vertex a_2; and the subgraph induced by the vertices colored red or green is a forest consisting of the path on the four vertices a_3, a_6, a_5, a_8 and the two isolated vertices a_1 and a_{10}.

Our goal here is to show how the ordering in a substitutable partition can be computed from an acyclic coloring of the adjacency graph $G(A)$ of the symmetric matrix A. First, it is easy to verify that in a group of columns in a substitutable partition induced by an acyclic coloring, each diagonal element is the only nonzero in its row. This is an immediate consequence of the fact that an acyclic coloring is a distance-1 coloring, for if a_{ij} is nonzero, vertices a_i and a_j are adjacent and are assigned different colors in a distance-1 coloring. Thus a_i is the only column with a nonzero in row i in the group of columns to which it belongs.

Next, consider off-diagonal elements a_{ij} with $i \neq j$, where the two elements a_{ij} and a_{ji} are identified due to symmetry. Each edge in $G(A)$ belongs to exactly one two-colored subgraph of $G(A)$, since its two endpoints are assigned distinct colors. Thus an acyclic coloring partitions the edges of the graph $G(A)$ into two-colored subgraphs, which are forests. We will solve for the matrix elements corresponding to the edges from each two-colored tree separately, and thus we need to prescribe the order only for a tree.

In a two-colored tree, every edge incident on a leaf vertex can be solved for immediately since it is the only nonzero in a row of the group of columns to which its parent vertex belongs. As an example, consider the red-blue forest induced by the vertices $\{a_1, a_2, a_3, a_4, a_5, a_7, a_9\}$ in Figure 6.1. The relevant nonzeros in the two groups of columns $\{a_1, a_3, a_5\}$ and $\{a_2, a_4, a_7, a_9\}$ in the partition induced by the acyclic coloring are as follows:

$$\begin{pmatrix} a_{21} & a_{23} & a_{25} \\ & a_{43} & \\ & \mathbf{a_{63}} & \mathbf{a_{65}} \\ a_{71} & & \\ & & \mathbf{a_{85}} \end{pmatrix}; \quad \begin{pmatrix} a_{12} & & a_{17} & \\ a_{32} & a_{34} & & \\ a_{52} & & & \\ & & & \mathbf{a_{69}} \\ & & \mathbf{a_{87}} & \mathbf{a_{89}} \\ & \mathbf{a_{10,4}} & & \mathbf{a_{10,9}} \end{pmatrix}.$$

Note that we have not shown the diagonal nonzeros in each column. Furthermore, the elements shown in bold will be evaluated when we consider the other two-colored forests; the bold entries in the submatrix on the left will be evaluated from the red-green forest, and the bold entries in the submatrix on the right will be evaluated from the blue-green forest.

Consider the nontrivial tree in the red-blue forest. In this tree, the vertex a_7 is a leaf, and the edge (a_7, a_1) can be computed from column a_1 since a_{71} is the only nonzero in row 7 in the columns colored red, the set $\{a_1, a_3, a_5\}$. The vertex a_4 is also a leaf, and the edge (a_4, a_3) can be computed from column a_3 since a_{43} is the only nonzero in row 4 in the columns colored red. Likewise, vertex a_5 is a leaf, and the edge (a_5, a_2) can be computed from column a_2 since a_{52} is the only nonzero in row 5 in the columns colored blue, the set $\{a_2, a_4, a_7, a_9\}$.

Furthermore, once the edges incident on a leaf have been evaluated, they can be deleted from the tree to create new leaves. The process can be repeated to solve for edges incident on the new leaf vertices, by using values computed for the leaf edges from earlier steps. The deletion of edges corresponds to substitution of already computed quantities in a triangular system of equations. In our illustration, once edges (a_7, a_1), (a_4, a_3), and (a_5, a_2) have been evaluated and deleted, the path a_1, a_2, a_3 remains. In this path, vertices a_1 and a_3 are leaves. In the columns colored blue, row 1 has two nonzeros, a_{12} and a_{17}; since the latter is known from the evaluation of a_{71}, we can compute the element a_{12} by substitution. Finally, the last edge in the tree, edge (a_2, a_3), can be evaluated by using row 1 of the columns colored red, by substituting the known values for a_{21} and a_{25}, or by using row 2 of the columns colored blue, by substituting the known value for a_{34}.

To summarize, an ordering from which matrix elements can be evaluated follows naturally from an acyclic coloring by considering the partition of the graph into two-colored trees. In each tree, the edges incident on a leaf vertex can be evaluated directly; subsequently, as the evaluated edges are deleted, edges incident on new leaf vertices could be evaluated in a substitution method. This process can be repeated until the tree becomes empty and every edge in the tree has been evaluated.

Coleman and Cai [27] established the connection between acyclic coloring and the computation of a symmetric matrix using a substitution method. Acyclic coloring had been studied earlier by Grünbaum [56] in a different context. Further information on acyclic coloring with pointers to references is included in section 11.4.

THEOREM 6.4 (Coleman and Cai [27]). *Let A be a symmetric matrix with nonzero diagonal elements and let $G(A) = (V, E)$ be its adjacency graph representation. A mapping ϕ is an acyclic coloring of $G(A)$ if and only if ϕ induces a substitutable partition of the columns of A.*

Proof. Assume that ϕ is an acyclic coloring of the graph $G(A)$. We claim that ϕ induces a substitutable partition of the columns of A. This is immediate from the earlier discussion, since an acyclic coloring partitions the edges of $G(A)$ into two-colored forests, and the edges in each two-colored tree can be computed by the process of computing edges incident on leaf vertices.

Now we prove the converse. Assume that we are given a substitutable partition of the columns $C = \{C_1, C_2, \ldots, C_p\}$ of the symmetric matrix A. Define a mapping $\phi(a_i) = k$ if $a_i \in C_k$ for $1 \leq i \leq p$. We will show that ϕ corresponds to an acyclic coloring of the vertices of the adjacency graph $G(A)$.

To arrive at a contradiction, assume that ϕ is not an acyclic coloring. Then either it is not a distance-1 coloring, or the graph $G(A)$ has a cycle in which the vertices are assigned only two colors.

If ϕ is not a distance-1 coloring, then there are two adjacent vertices, say, a_i and a_j in the graph $G(A)$ with the same color. Then these columns belong to the same group in C. The columns a_i and a_j contain the following 2×2 submatrix:

$$\begin{pmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{pmatrix}.$$

These three distinct nonzero elements are not present in any other column and hence cannot be evaluated from any other group. But their values cannot be computed by solving a triangular system of equations since there are two unknown elements in each of the two rows. This contradicts the assumption that C is a substitutable partition.

Now consider the case when ϕ induces a two-colored cycle in the graph $G(A)$. By assumption, the elements corresponding to the edges of the cycle can be computed from the substitutable partition of A. Let (a_i, a_j) denote the first edge computed from the two-colored cycle. Let a_h denote the neighbor of a_i distinct from a_j in the cycle, and similarly let a_k denote the neighbor of a_j distinct from a_i in the cycle. Then, by choice of the edge (a_i, a_j), the edges (a_h, a_i) and (a_j, a_k) have not been evaluated yet. Thus the pair of nonzeros a_{hi} and a_{ij} (and also a_{ij} and a_{jk}) cannot be ordered to satisfy the requirements of a substitutable partition given in Definition 6.1. This contradicts the assumption that C is a substitutable partition. □

In view of Theorem 6.4, Problem 6.2 is equivalent to the following graph problem.

PROBLEM 6.5. *Given the adjacency graph $G(A) = (V, E)$ representing the sparsity structure of a symmetric matrix A with nonzero diagonal elements, find an acyclic coloring of $G(A)$ that uses the fewest colors.*

6.1.3. An Acyclic Coloring Algorithm. Coleman and Cai [27] proved that the problem of finding an acyclic coloring of a graph that uses the fewest colors is NP-hard. Recently, we developed a new efficient heuristic algorithm for the problem. Here we briefly mention the main idea in the algorithm; for a more detailed discussion see [48].

Our algorithm relies on the fact that in a graph where the vertices have been colored satisfying the conditions of acyclic coloring, every subgraph induced by a set of vertices assigned any two colors is a forest. The union of *all* two-colored forests in a partially acyclically colored graph is a collection of edge-disjoint trees. Each tree in the collection is dynamic in the sense that edges may be added to it as the algorithm proceeds. Based on these observations, our algorithm uses the *disjoint-set data structure* to maintain the collection of trees in an efficient way. In our context, a set in the disjoint-set data structure corresponds to a two-colored tree and an element of a set corresponds to an edge. The two important operations supported by the disjoint-set data structure are union and find.

The algorithm iterates over the vertices in the set V in some order. In each step i, $1 \leq i \leq |V|$, it ensures that the color chosen for vertex v_i (1) is distinct from each of the colors of the distance-1 neighbors of v_i, and (2) does not lead to a cycle in any one of the current two-colored trees. The first requirement is easy to enforce: the colors used by the distance-1 neighbors of vertex v_i are excluded from the set of colors allowed for vertex v_i. Since we maintain a collection of two-colored trees, the second requirement reduces to checking whether vertex v_i is connected to at least two same-colored vertices in a single two-colored tree. These checks can be done systematically for all the trees incident on v_i in order to exclude additional colors from the set of colors allowed to vertex v_i. Once the colors that are not allowed for vertex v_i are determined, the smallest allowable color is chosen and assigned to v_i. The computational work involved in coloring the vertex v_i is proportional to $d_2(v_i)$. The

total number of **find** operations in the algorithm is bounded by $f = \sum_{v \in V} d_2(v)$, while the number of **union** operations is at most $|E|$. Thus the overall time complexity of the algorithm is $O(\alpha(f, |E|) \cdot |V|\bar{\delta}_2)$, where α is the functional inverse of Ackermann's function. Recently, we used the underlying technique in this algorithm, exploiting the structure of two-colored induced subgraphs, to design an $O(|V|\bar{\delta}_2)$-time star coloring algorithm. See the report [48] for a detailed discussion of these algorithms.

Earlier, Coleman and Cai [27] suggested an algorithm for the acyclic coloring problem. The idea in their algorithm is to first transform a given graph $G = (V, E)$ to a "completed" graph $G' = (V, E')$ by adding edges in such a way that every cycle in G includes a triangle in G', and then use a known distance-1 coloring heuristic on G'. The construction of G' is done in the following way. Initially set E' to be the same as E. Consider an ordering (numbering) $\pi : V \leftrightarrow \{1, 2, \ldots, n\}$ of the vertices. Process the vertices in the order given by π. In each iteration i, if the vertex $w = \pi^{-1}(i)$ is such that there exists a path v, w, x in G where $\pi(w) > \max\{\pi(v), \pi(x)\}$, then add the edge (v, x) to E'. It is clear that when G' is constructed in this manner, a distance-1 coloring of G gives a valid acyclic coloring of G. However, an *optimal* distance-1 coloring of G' does not imply an optimal acyclic coloring of G. In fact, the algorithm of Coleman and Cai is an algorithm for a more constrained coloring problem that Coleman and Moré [31] have called triangular coloring. (A vertex coloring ϕ of a graph G is *triangular* if there exists an ordering π of the vertices such that (1) ϕ is a distance-1 coloring of G, and (2) in every path v, w, x in G, the vertices v and x receive different colors whenever $\pi(w) > \max\{\pi(v), \pi(x)\}$.)

The graph G' used in the algorithm described above depends critically on the chosen ordering π. Coleman and Moré [31] showed that a *smallest last* vertex ordering is particularly well-suited for minimizing the number of colors required to distance-1 color the graph G'. For a discussion of the smallest last ordering and its relationship to various graph parameters, see section 11.

The graph G' has interesting relationships with *fill* graphs in sparse matrix factorization. Our first observation is that the graph G' is a subgraph of a *chordal completion* $G^+ = (V, E \cup F)$ of G. A graph is *chordal* if every cycle on four or more vertices has a chord, an edge connecting two nonconsecutive vertices on the cycle. Computing a chordal completion of a graph in which the number of fill edges $|F|$ is minimized is NP-hard [124]. This graph problem is a well-known model for sparse Cholesky factorization. The process through which G' is computed resembles, but is not the same as, the *elimination game*, a process through which G^+ is computed. The difference is that in the former case a fill edge is added between two lower numbered vertices whenever these vertices have a common higher numbered neighbor in the *original* graph G.

We can also characterize the edges in the graph G' in terms of fill in a sparse incomplete LU factorization. In a level-based incomplete LU factorization, each of the edges in the original graph are assigned level 0, and the fill edges are assigned higher values of levels. In the *sum rule* of assigning levels to fill edges, when a fill edge (v_j, v_k) is created by a pair of edges (v_i, v_j) and (v_i, v_k), the level of the fill edge is one more than the sum of the levels of the two causative edges. In this case, since the causative edges belong to the original graph G, they have level values equal to 0, and hence the fill edge (v_j, v_k) has level 1. Thus the added edges $E' \setminus E$ in G' are exactly the fill edges in a level-based sparse incomplete LU factorization corresponding to level 1, i.e., ILU(1).

Hysom and Pothen [70] proved a theorem characterizing fill in an incomplete factorization. A *fill path* between two vertices v_i and v_j is a path in the original graph

ALGORITHM 6.1. A scheme for solving the acyclic bicoloring problem.

procedure ACYCLICBICOLORINGSCHEME($G_b = (V_1, V_2, E)$)
1. Find a suitable vertex cover C in G_b
2. Assign the vertices in the set $I = (V_1 \cup V_2) \setminus C$ the color 0
3. Color the vertices in C such that the result is an acyclic bicoloring of G_b

end procedure

G joining the two vertices such that all of its interior vertices are numbered less than v_i and v_j. An edge (v_i, v_j) is a fill edge of level ℓ if and only if the shortest fill path joining v_i and v_j in the graph G has length $\ell + 1$ edges [70].

6.2. Computing the Jacobian. In estimating a nonsymmetric matrix using a substitution method, the requirement on the bidirectional partition can be relaxed so as to obtain fewer groups compared to a partition in a direct method.

6.2.1. The Matrix Partitioning Problem. We state the following definition, due to Coleman and Verma [32], to subsequently describe the fifth matrix partitioning problem of our concern.

DEFINITION 6.6. *A bidirectional partition* (Π_C, Π_R) *of a matrix* A *is* substitutable *if there exists an ordering on the elements of* A *such that for every nonzero element* a_{ij}*, either* (1) *column* a_j *is in a group where all nonzeros in row* r_i*, from other columns in the group, are ordered before* a_{ij}*, or* (2) *row* r_i *is in a group where all the nonzeros in column* a_j*, from other rows in the group, are ordered before* a_{ij}*.*

PROBLEM 6.7. *Given the sparsity structure of an* $m \times n$ *matrix* A*, find a substitutable bidirectional partition* (Π_C, Π_R) *of* A *such that* $|\Pi_C| + |\Pi_R|$ *is minimized.*

6.2.2. A Graph Coloring Formulation. The relationship between bicoloring and bidirectional partition, established by Theorem 5.6, coupled with that between acyclic coloring and substitutable partition, established by Theorem 6.4, suggests that "acyclic bicoloring" might be the right graph model for Problem 6.7. Coleman and Verma [32] showed that this was indeed the case.

DEFINITION 6.8. *Let* $G_b = (V_1, V_2, E)$ *be a bipartite graph. A mapping* $\phi : [V_1, V_2] \to \{0, 1, \ldots, p\}$ *is an* acyclic bicoloring *of* G_b *if the following conditions hold:*
1. *If* $u \in V_1$ *and* $v \in V_2$*, then* $\phi(u) \neq \phi(v)$ *or* $\phi(u) = \phi(v) = 0$*.*
2. *If* $(u, v) \in E$*, then* $\phi(u) \neq 0$ *or* $\phi(v) \neq 0$*.*
3. *If vertices* u *and* v *are adjacent to a vertex* w *with* $\phi(w) = 0$*, then* $\phi(u) \neq \phi(v)$*.*
4. *Every cycle uses at least three colors.*

THEOREM 6.9 (Coleman and Verma [32]). *Let* A *be an* $m \times n$ *matrix and* $G_b(A) = (V_1, V_2, E)$ *be its bipartite graph. The mapping* $\phi : [V_1, V_2] \to \{0, 1, \ldots, p\}$ *is an acyclic bicoloring of* $G_b(A)$ *if and only if* ϕ *induces a substitutable bidirectional partition* (Π_C, Π_R) *of* A*.*

By Theorem 6.9, the following coloring problem is equivalent to Problem 6.7.

PROBLEM 6.10. *Given the bipartite graph* $G_b(A) = (V_1, V_2, E)$ *representing the sparsity structure of an* $m \times n$ *matrix* A*, find an acyclic bicoloring of* $G_b(A)$ *that uses the fewest colors.*

6.2.3. An Acyclic Bicoloring Algorithm. Observation 5.8 suggests the scheme ACYCLICBICOLORINGSCHEME outlined in Algorithm 6.1 for solving the acyclic bicoloring problem. Step 3 of ACYCLICBICOLORINGSCHEME can be done using an appropriate adaptation of the acyclic coloring algorithm sketched in section 6.1.3.

7. Interrelationships among the Coloring Problems. In this short section, we expose the interrelationships among the various coloring variants introduced thus far in this paper. The relationships reveal that distance-2 coloring is the most *general* variant.

7.1. Chromatic Numbers. The conditions required by distance-1 coloring, acyclic coloring, star coloring, acyclic bicoloring, star bicoloring, and distance-2 coloring imply the relationships among the respective chromatic numbers stated in Theorems 7.1 and 7.2. The two theorems are due to Coleman and Cai [27] and Coleman and Verma [32].

Recall that the distance-k chromatic number of a graph G is denoted by $\chi_k(G)$, and the least number of colors required for a partial distance-2 coloring of a bipartite graph $G_b = (V_1, V_2, E)$ on V_i $(i = 1, 2)$ is denoted by $\chi_2(G_b, V_i)$. Let the chromatic number for acyclic and star coloring of a graph G be denoted by $\chi_a(G)$ and $\chi_s(G)$, respectively. Further, let the chromatic number for acyclic and star bicoloring of a bipartite graph G_b be denoted by $\chi_{ab}(G_b)$ and $\chi_{sb}(G_b)$, respectively.

THEOREM 7.1. *For every graph $G = (V, E)$,*

$$\chi_1(G) \leq \chi_a(G) \leq \chi_s(G) \leq \chi_2(G) = \chi_1(G^2).$$

Proof. Regarding the first three inequalities, observe that a distance-2 coloring is a star coloring; a star coloring is an acyclic coloring; and an acyclic coloring is a distance-1 coloring. The last equality holds since a distance-2 coloring of G is equivalent to a distance-1 coloring of G^2. □

THEOREM 7.2. *For every bipartite graph $G_b = (V_1, V_2, E)$,*

$$\chi_{ab}(G_b) \leq \chi_{sb}(G_b) \leq \min\{\chi_2(G_b, V_1), \chi_2(G_b, V_2)\}.$$

Proof. The first inequality is obvious. For the second inequality, observe that a partial distance-2 coloring on V_2 is a valid star bicoloring of the bipartite graph G_b where all the vertices in V_1 are restricted to be colored with color zero. A similar argument, with the roles of V_1 and V_2 interchanged, can be used to complete the proof. □

In the context of efficient derivative matrix computation, the implication of Theorem 7.2 is that an optimal bidirectional partition, irrespective of the structure of the matrix, yields at most as many groups as an optimal unidirectional partition, and hence potentially results in a more efficient computation.

Theorems 7.1 and 7.2 show that distance-2 coloring is an archetypal model in the computation of Jacobian and Hessian matrices.

Distance-2 coloring also has applications other than derivative matrix computation. Examples include channel assignment [84] and facility location problems (see Chapter 5 of the book by Vazirani [117]).

7.2. Two-Colored Induced Subgraphs. As discussed in earlier sections, the names for star and acyclic coloring are derived from the structure of two-colored induced subgraphs in the respective colorings. In this section we revisit this issue and extend the observation to the distance-1 and distance-2 coloring cases.

Let $G = (V, E)$ be a graph and $\phi : V \to \{1, 2, \ldots, p\}$ be a variant of vertex coloring. Let $H_{2colors} \subseteq G$ denote a subgraph of G induced by the union of any two color classes in ϕ. In Table 7.1 we summarize our observations regarding the characterization of $H_{2colors}$ in the cases where ϕ is a distance-1, an acyclic, a star, and a distance-2 coloring. The reader will find the illustration in Figure 7.1 helpful

Table 7.1 *Characterization of the subgraph induced by the union of any two color classes.*

ϕ	$H_{2colors}$
Distance-1 coloring	A bipartite graph
Acyclic coloring	A collection of trees
Star coloring	A collection of stars
Distance-2 coloring	A distance-2 matching

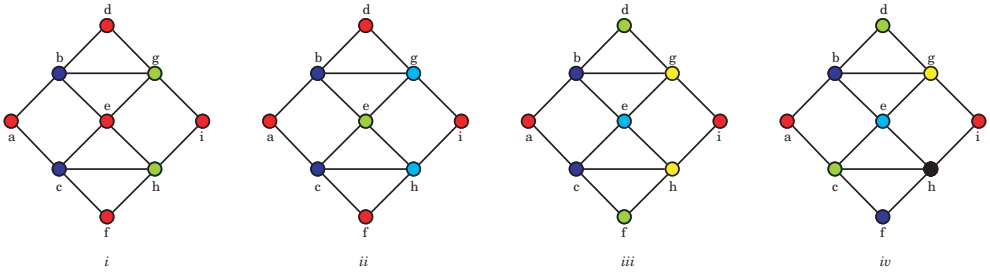

Fig. 7.1 *(Left to right) A distance-*1*, an acyclic, a star, and a distance-*2 *coloring of a graph.*

Table 7.2 *The color classes in each of the four colorings depicted in Figure* 7.1.

	i	ii	iii	iv
Class 1	$\{a, d, e, f, i\}$	$\{a, d, f, i\}$	$\{a, i\}$	$\{a, i\}$
Class 2	$\{b, c\}$	$\{b, c\}$	$\{b, c\}$	$\{b, f\}$
Class 3	$\{g, h\}$	$\{e\}$	$\{d, f\}$	$\{c, d\}$
Class 4		$\{g, h\}$	$\{e\}$	$\{e\}$
Class 5			$\{g, h\}$	$\{g\}$
Class 6				$\{h\}$

in following this discussion; parts (i) through (iv) depict an example of each of the respective coloring variants. For convenience, Table 7.2 lists the color classes in each of the four cases of Figure 7.1.

It is obvious that ϕ is a distance-1 coloring if and only if $H_{2colors}$ is a (not necessarily connected) bipartite graph.

Since ϕ in Table 7.1 gets progressively restricted as one goes down the column, correspondingly, $H_{2colors}$ becomes progressively restricted (and less connected). Acyclic coloring is named as such precisely because the corresponding $H_{2colors}$ is a collection of trees (an acyclic graph). Similarly, star coloring owes its name to the fact that the corresponding $H_{2colors}$ is a collection of stars [41]. It is easy to see that a connected two-colored induced subgraph here ought to be a star, for otherwise there would exist a two-colored path on four vertices, violating a condition of star coloring.

In the case where ϕ is a distance-2 coloring, we note that $H_{2colors}$ is a restricted matching. In the usual sense, a *matching* in a given graph $G = (V, E)$ is a set $M \subseteq E$ of edges with no shared endpoints. In other words, in a connected graph G, a pair of edges in a matching M is at least *one* edge apart in G. The matching $H_{2colors}$ in the last row of Table 7.1 is such that a pair of edges in $H_{2colors}$ is apart by a path of length at least *two* edges in G. We call this a *distance-*2 *matching*. The subgraph induced by the vertices $\{b, c, d, f\}$ in part (iv) is an example of a distance-2 matching, since it consists of the two edges (b, d) and (c, f), which are apart from each other by paths of length two or greater.

As a corollary, we observe a relationship between distance-2 vertex coloring and a restricted variant of edge coloring. In the standard usage, an *edge coloring* of a graph is an assignment of colors to its edges such that every pair of edges sharing an endpoint receives different colors. Such an edge coloring clearly partitions the edge set into matchings. A *distance*-2 *edge coloring* is an assignment of colors to edges such that every pair of edges that either shares an endpoint or is separated by one other edge is assigned different colors. Consider now the edge coloring ϕ' derived from a distance-2 vertex coloring ϕ in the following way: assign each edge a color obtained by "mixing" the colors of its two endpoints; i.e., for each $e = (u, v)$, $\phi'(e) = \phi(u) \odot \phi(v)$. Here, $x \odot y$ denotes the number obtained by *concatenating* the positive integers x and y; e.g., $2 \odot 3$ is 23. It is evident that the edge coloring ϕ' obtained in this manner is a distance-2 edge coloring and hence partitions the edge set into distance-2 matchings.

8. Partial Matrix Computation. In many large-scale optimization contexts, the Jacobian or the Hessian is formed only for preconditioning purposes, and only a *subset* of the matrix elements needs to be computed. Computing a good preconditioner is critical for fast convergence to a solution. A recent survey article by Knoll and Keyes [79] discusses various applications where the "Jacobian-free Newton–Krylov" method is used. A basic ingredient of this method is an approximate computation of some elements of the Jacobian. Also, there are examples in which only certain elements of the Hessian need to be updated in an iterative procedure, since the other elements do not change in value [11].

In this section we develop graph coloring formulations of partitioning problems that arise in the computation of a *specified subset* of the nonzero entries of a matrix. We call this *partial* matrix computation as opposed to *full* matrix computation, the case in which all nonzero entries are required to be determined.

The coloring formulations in this section are new and more sophisticated than the coloring formulations in full matrix computation. In a coloring formulation of partial matrix estimation, one may be tempted to think that the vertices corresponding to the nonzeros outside the set of required elements might as well be ignored. However, such vertices still need to be considered since they could interfere with the computation of the required elements. The motivation for developing the new graph formulations is that efficient partial matrix computation can be used to further reduce the number of colors needed to compute the required elements. For example, if only the diagonal elements of a Hessian are needed, then we need only a *distance*-1 *coloring* of the adjacency graph, rather than a star coloring, the coloring required for full matrix computation.

Similar to the bicoloring cases discussed in earlier sections, in the colorings defined here we allow a vertex to be assigned the "neutral" color zero. A vertex with color zero signifies the fact that a column or a row that corresponds to the vertex is not used to compute any element in that column or row.

The rest of this section is organized in three parts. Each part deals with a scenario defined by the kind of matrix under consideration (Jacobian or Hessian) and the type of partition employed (unidirectional or bidirectional). In each case, the required entries are assumed to be determined using a direct method. The problems that correspond to computation via substitution are not studied in this work.

8.1. Unidirectional Computation of the Jacobian. Let A be an $m \times n$ nonsymmetric matrix, and let S denote the set of nonzero elements of A required to be computed. A partition of a subset of the columns of A is *structurally orthogonal when*

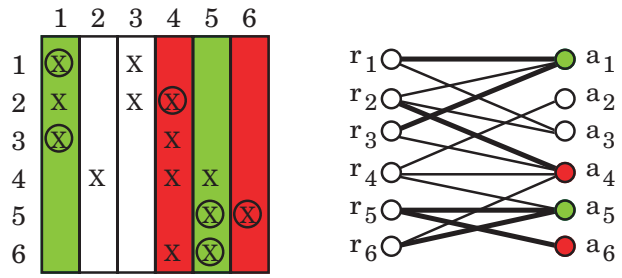

Fig. 8.1 *Partial estimation of a nonsymmetric matrix and its formulation as a restricted distance-2 coloring in the associated bipartite graph. The required matrix entries are encircled and the corresponding edges in the graph are shown in bold. The illustration uses two colors: columns a_1 and a_5 are green, and columns a_4 and a_6 are red.*

restricted to S if for every $a_{ij} \in S$, column a_j is included in some group that contains no other column with a nonzero in row r_i. (The latter nonzero could be inside or outside S.) Such a partition enables a direct determination of the elements of S.

Let $G_b(A) = (V_1, V_2, E)$ be the bipartite graph of A, and let $E_S \subseteq E$ correspond to the elements of S. A mapping $\phi : V_2 \rightarrow \{0, 1, \ldots, p\}$ is a *distance-2 coloring of* G_b *when restricted to* E_S if the following conditions hold for every edge $(v, w) \in E_S$, where $v \in V_1$, $w \in V_2$:

1. $\phi(w) \neq 0$, and
2. for every path (u, v, w), $\phi(u) \neq \phi(w)$. (Here the edge (u, v) may or may not belong to E_S.)

Theorem 8.1 below shows the equivalence between the restricted partitioning and coloring variants introduced above. See Figure 8.1 for an illustration; in the figure, the nonzero matrix entries required to be computed are encircled and the corresponding edges in the bipartite graph are shown in bold; columns a_1 and a_5 are colored green, and columns a_4 and a_6 are colored red.

THEOREM 8.1. *The mapping ϕ is a distance-2 coloring of $G_b(A)$ when restricted to E_S if and only if ϕ induces a structurally orthogonal column partition when restricted to S.*

Proof. Assume that ϕ is a distance-2 coloring of $G_b(A)$ when restricted to E_S. Let p be the number of colors used. We show that the groups $\{C_1, \ldots, C_p\}$ where $C_\alpha = \{a_j : \phi(a_j) = \alpha\}$, $1 \leq \alpha \leq p$, constitute a structurally orthogonal column partition when restricted to S. First, observe that by condition 1, for every $a_{ij} \in S$ (i.e., $(r_i, a_j) \in E_S$), $\phi(a_j) \neq 0$. Thus column a_j belongs to group $C_{\phi(a_j)}$ and hence is involved in the partition. Assume now that the partition induced by the coloring is not structurally orthogonal when restricted to S. This occurs only if there exist nonzero elements a_{ij} and a_{ik}, $j \neq k$, such that $a_{ij} \in S$ and both a_j and a_k belong to group $C_{\alpha'}$ for some α', $1 \leq \alpha' \leq p$. But this contradicts condition 2, and hence cannot occur.

Conversely, assume that the partition $C = \{C_1, \ldots, C_p\}$ is structurally orthogonal when restricted to S. Construct a coloring ϕ of $G_b(A)$ as follows: $\phi(a_j) = \alpha$ if $a_j \in C_\alpha$, and $\phi(a_j) = 0$ if a_j does not belong to any group in C. We claim that ϕ is a distance-2 coloring of $G_b(A)$ when restricted to E_S. Each vertex in V_2 incident on an edge in E_S corresponds to a column with an entry in S, and hence gets a nonzero color. Thus ϕ satisfies condition 1. Consider any path (a_j, r_i, a_k) where $(r_i, a_j) \in E_S$. Note that such a path in $G_b(A)$ exists whenever entries a_{ij} and a_{ik} are nonzero. Structural

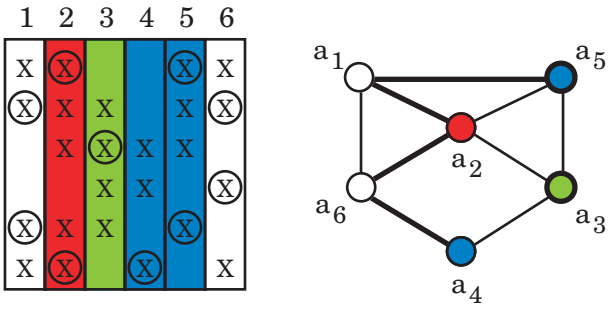

Fig. 8.2 *Partial estimation of a symmetric matrix and its formulation as a restricted star coloring in the adjacency graph. The required matrix entries are encircled, and edges of the graph corresponding to required off-diagonal matrix entries and vertices "corresponding" to required diagonal matrix entries are shown in bold. The illustration uses three colors: column a_2 is red, a_3 is green, and both a_4 and a_5 are blue.*

orthogonality when restricted to S implies that column a_k cannot be in the same group as column a_j. Thus, by construction, $\phi(a_j) \neq \phi(a_k)$, satisfying condition 2. □

8.2. Computing the Hessian. Let A be a symmetric matrix with nonzero diagonal elements, and let S denote the set of nonzero elements of A required to be computed. A partition of a subset of the columns of A is *symmetrically orthogonal when restricted to* S if for every $a_{ij} \in S$ at least one of the following two partition conditions are met:

1. The group containing column a_j has no other column with a nonzero in row r_i.
2. The group containing column a_i has no other column with a nonzero in row r_j.

Such a partition enables a direct determination of the elements of S.

Let $G(A) = (V, E)$ be the adjacency graph of A; let $S_{od} \subseteq E$ correspond to the off-diagonal elements in S; and let S_d correspond to the diagonal elements in S, i.e., $S_d = \{(u, u) : u \in U\}$ where $U \subseteq V$. Let $E_S = S_{od} \cup S_d$. A mapping $\phi : V \to \{0, 1, 2, \ldots, p\}$ is a *star coloring of* G *when restricted to* E_S if the following coloring conditions hold:

1. For every $(u, u) \in S_d$,
   1.1. $\phi(u) \neq 0$, and
   1.2. for every $(u, v) \in E$, $\phi(u) \neq \phi(v)$.
2. For every $(v, w) \in S_{od}$,
   2.1. $\phi(v) \neq \phi(w)$, and
   2.2. at least one of the following two conditions holds:
      2.2.1. $\phi(v) \neq 0$ and for every path (v, w, x), $\phi(v) \neq \phi(x)$, or
      2.2.2. $\phi(w) \neq 0$ and for every path (u, v, w), $\phi(u) \neq \phi(w)$.

The following theorem states the equivalence between the restricted partitioning and coloring variants introduced above. See Figure 8.2 for an illustration; in the figure, the matrix entries required to be computed are encircled, edges in the adjacency graph corresponding to the required off-diagonal entries are shown in bold, and column vertices containing a required diagonal element are shown in bold. In the illustration, column a_2 is colored red, a_3 is green, and a_4 and a_5 are blue.

THEOREM 8.2. *The mapping ϕ is a star coloring of $G(A)$ when restricted to E_S if and only if ϕ induces a symmetrically orthogonal partition when restricted to S.*

Proof. Assume that ϕ is a star coloring of $G(A)$ when restricted to E_S. Let the number of colors used by ϕ be p. We show that the groups $\{C_1, \ldots, C_p\}$ where $C_\alpha = \{a_j : \phi(a_j) = \alpha\}$, $1 \leq \alpha \leq p$, constitute a symmetrically orthogonal partition when restricted to S.

By coloring conditions 1.1 and 2.1, for every $a_{ij} \in S$ (i.e., $(a_i, a_j) \in E_S$), at least one of the vertices a_i or a_j has a nonzero color and hence the column is involved in the partition $\{C_1, \ldots, C_p\}$. Let $a_{ij} \in S$ be a diagonal entry ($i = j$). Then coloring conditions 1.1 and 1.2 ensure that $\phi(a_i) \neq 0$ and that $\phi(a_i) \neq \phi(a_k)$ for every $(a_i, a_k) \in E$. Thus, by construction, column a_i belongs to group $C_{\phi(a_i)}$ and no column a_k, where $i \neq k$ and $a_{ik} \neq 0$, is in $C_{\phi(a_i)}$. This clearly satisfies the partition conditions.

Let $a_{ij} \in S$ now be an off-diagonal entry ($i \neq j$). Assume without loss of generality that $\phi(a_j) \neq 0$. By coloring condition 2.1, $\phi(a_i) \neq \phi(a_j)$. By coloring condition 2.2.2, there is no path (a_h, a_i, a_j) in $G(A)$, for any $h \neq i, j$, such that $\phi(a_h) = \phi(a_j)$. The last two statements together imply that column a_j belongs to group $C_{\phi(a_j)}$ and that no column a_h, where $h \neq j$ and $a_{ih} \neq 0$, is in $C_{\phi(a_j)}$. This satisfies the first partition condition. A similar argument applies to the case where $\phi(a_i) \neq 0$, which implies the satisfaction of the alternate partition condition.

To prove the converse, assume that the partition $C = \{C_1, \ldots, C_p\}$ is symmetrically orthogonal when restricted to S. Construct a coloring ϕ of $G(A)$ as follows. Define $\phi(a_j) = \alpha$ if $a_j \in C_\alpha$, and $\phi(a_j) = 0$ if a_j does not belong to any group in C. We claim that ϕ is a star coloring of $G(A)$ when restricted to E_S.

Consider a diagonal element $a_{ii} \in S$. The partition conditions ensure that column a_i is in some group $C_{\alpha'}$ and that there is no column $a_k \in C_{\alpha'}$, $i \neq k$, such that $a_{ik} \neq 0$. Thus, by construction, $\phi(a_i) \neq 0$ and $\phi(a_i) \neq \phi(a_k)$ for every $(a_i, a_k) \in E$, satisfying coloring condition 1.

Consider now the case where $a_{ij} \in S$ is an off-diagonal element. First, observe that since all diagonal elements are nonzero, a_i and a_j cannot belong to the same group. Thus $\phi(a_i) \neq \phi(a_j)$, satisfying coloring condition 2.1. Second, observe that there are two possibilities by which the partitioning conditions can be satisfied. We consider only one of these; the second can be treated in a similar manner. Suppose column a_j belongs to some group $C_{\alpha'}$ and that there is no other column $a_h \in C_{\alpha'}$, $h \neq j$, such that $a_{ih} \neq 0$. Thus, by construction, $\phi(a_j) \neq 0$ and $\phi(a_h) \neq \phi(a_j)$ for every path (a_h, a_i, a_j) in $G(A)$, satisfying coloring condition 2.2.2. ∎

A special case of Theorem 8.2 is the problem of estimating *only* the diagonal elements of A, i.e., $S_d = \{(v, v) : v \in V\}$ and $S_{od} = \emptyset$. For this problem, coloring conditions 1.1 and 1.2 are the only applicable conditions, and they imply that a distance-1 coloring of $G(A)$ suffices.

8.3. Bidirectional Computation of the Jacobian. Let A be an $m \times n$ nonsymmetric matrix, and let S denote the set of nonzero elements of A required to be computed. A bidirectional partition (Π_C, Π_R) of a subset of the columns and rows of A is *structurally orthogonal when restricted to* S if for every $a_{ij} \in S$ at least one of the following two partition conditions are met:

1. The group containing column a_j has no other column with a nonzero in row r_i.
2. The group containing row r_i has no other row with a nonzero in column a_j.

Such a partition enables a direct determination of the elements of S.

Let $G_b(A) = (V_1, V_2, E)$ be the bipartite graph of A, and let $E_S \subseteq E$ correspond to the elements in S. A mapping $\phi : [V_1, V_2] \to \{0, 1, \ldots, p\}$ is a *star bicoloring of* G_b *when restricted to* E_S if the following coloring conditions are met:

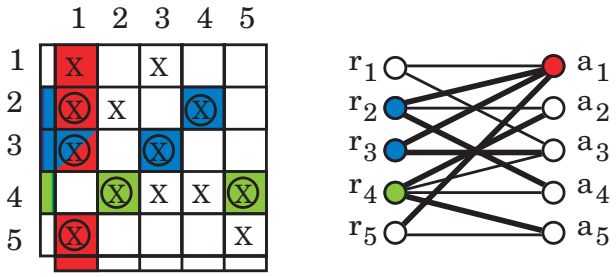

Fig. 8.3 *Partial estimation of a nonsymmetric matrix via bidirectional partition and its formulation as a restricted star bicoloring in the bipartite graph. The required matrix entries are encircled and the corresponding edges in the graph are shown in bold. The illustration uses three colors: both r_2 and r_3 are blue, r_4 is green, and a_1 is red.*

1. Vertices in V_1 and V_2 receive disjoint colors, except for color 0;
   i.e., for every $u \in V_1$ and $v \in V_2$, either $\phi(u) \neq \phi(v)$ or $\phi(u) = \phi(v) = 0$.
2. At least one endpoint of an edge in E_S receives a nonzero color;
   i.e., for every $(v, w) \in E_S$, $\phi(v) \neq 0$ or $\phi(w) \neq 0$.
3. For every edge $(v, w) \in E_S$,
   3.1. if $\phi(v) = 0$, then, for every path (u, v, w), $\phi(u) \neq \phi(w)$;
   3.2. if $\phi(w) = 0$, then, for every path (v, w, x), $\phi(v) \neq \phi(x)$;
   3.3. if $\phi(v) \neq 0$ and $\phi(w) \neq 0$, then for every path (u, v, w, x), either $\phi(u) \neq \phi(w)$ or $\phi(v) \neq \phi(x)$.

The following theorem establishes the equivalence between the restricted partitioning and coloring notions introduced above. See Figure 8.3 for an illustration.

THEOREM 8.3. *The mapping ϕ is a star bicoloring of G_b when restricted to E_S if and only if ϕ induces a structurally orthogonal bidirectional partition when restricted to S.*

Proof. Let the construction of a partition given a coloring, and vice versa, be done in a similar manner as in the proof of Theorem 8.1.

Assume that ϕ is a star bicoloring of $G_b(A)$ when restricted to E_S. Let the induced bidirectional partition be (Π_C, Π_R). Coloring condition 1 implies that (Π_C, Π_R) is a bidirectional partition. By condition 2, for every $a_{ij} \in S$, either $a_j \in \Pi_C$ or $r_i \in \Pi_R$ (or both). Assume now that (Π_C, Π_R) is not structurally orthogonal when restricted to S. This occurs only if one of the following cases holds for any nonzero $a_{ij} \in S$:

- $\phi(r_i) = 0, \phi(a_j) \neq 0$, and there exists a column a_k, where $j \neq k$ and $a_{ik} \neq 0$, such that $\phi(a_j) = \phi(a_k)$. But this contradicts coloring condition 3.1 and hence cannot occur.
- $\phi(a_j) = 0, \phi(r_i) \neq 0$, and there exists a row r_h, where $h \neq i$ and $a_{hj} \neq 0$, such that $\phi(r_h) = \phi(r_i)$. But this contradicts coloring condition 3.2 and hence cannot occur.
- $\phi(r_i) \neq 0, \phi(a_j) \neq 0$, and there exists a column a_k, where $j \neq k$ and $a_{ik} \neq 0$, and a row r_h, where $h \neq i$ and $a_{hj} \neq 0$, such that $\phi(a_j) = \phi(a_k)$ and $\phi(r_h) = \phi(r_i)$. But this contradicts coloring condition 3.3 and hence cannot occur.

Hence, the bipartition (Π_C, Π_R) is structurally orthogonal when restricted to S.

Conversely, assume that (Π_C, Π_R) is a structurally orthogonal bidirectional partition when restricted to S. Clearly, the constructed coloring ϕ satisfies conditions 1

and 2. To complete the proof, we show that ϕ also satisfies condition 3. Assume that ϕ violates condition 3. Then one of the following cases must hold:

- There exists a path (a_h, r_i, a_j) for some $(r_i, a_j) \in E_S$ such that $\phi(r_i) = 0$ and $\phi(a_h) = \phi(a_j)$. But this implies that element a_{ij} cannot be determined directly, contradicting the assumption that (Π_C, Π_R) is structurally orthogonal when restricted to S.
- There exists a path (r_i, a_j, r_k) for some $(r_i, a_j) \in E_S$ such that $\phi(a_j) = 0$ and $\phi(r_i) = \phi(r_k)$. Again this implies that element a_{ij} cannot be determined directly, a contradiction of our assumption.
- There exists a path (a_h, r_i, a_j, r_k) for some $(r_i, a_j) \in E_S$ such that $\phi(r_i) = \phi(r_k) \neq 0$ and $\phi(a_h) = \phi(a_j) \neq 0$. But this implies that element a_{ij} cannot be determined directly, contradicting the assumption. □

Algorithms. We have not developed specialized algorithms for the restricted coloring problems introduced in this section. However, we believe that the ideas used in the algorithms for the corresponding coloring problems in full matrix computation can be adapted to the restricted cases.

9. Hypergraph Coloring Formulations. In this section, we introduce yet another perspective concerning Problem 3.3, the partitioning problem that arises in the unidirectional computation of a Jacobian via a direct method. This view uses the notion of a hypergraph, which is a generalization of a graph. Besides being an interesting alternative perspective, a hypergraph formulation might be a worthwhile approach for modeling partial matrix computation problems for preconditioning purposes. However, hypergraph formulations are included here mainly for the sake of completeness.

9.1. Definitions. We begin by defining a few concepts that we need for our formulations. A *hypergraph* $H = (V, E)$ consists of a finite set V of *vertices* and a collection E of nonempty subsets of V called *hyperedges*. Note that the size of a hyperedge (i.e., the number of vertices in it) is not constrained to be two as in a graph. A hypergraph is called *r-uniform* if all of its hyperedges are of size r. Thus, a graph is a 2-uniform hypergraph.

We denote the vertex set and edge set of a hypergraph H by $V(H)$ and $E(H)$, respectively. When the hypergraph under consideration is clear from the context, we may use just V and E to refer to these sets.

The *line graph* $L(H)$ of hypergraph H is a graph where the vertices of $L(H)$ are the hyperedges of H and the edges of $L(H)$ are pairs of intersecting hyperedges of H.

Figure 9.1 (left) is a simple example that shows a hypergraph consisting of five vertices and three hyperedges; two hyperedges have size three and one hyperedge has size two. The figure on the right shows the corresponding line graph.

9.2. Hypergraph Coloring. Given a hypergraph $H = (V, E)$, a *strong vertex coloring* of H is a mapping $\phi : V \to \{1, 2, \ldots, p\}$ such that for every hyperedge $e \in E$ and every pair of vertices $\{u, v\} \subseteq e$, $\phi(u) \neq \phi(v)$. In words, we require that the vertices in every hyperedge be assigned distinct colors in a strong vertex coloring of a hypergraph. The more common variant of hypergraph coloring, called *weak vertex coloring*, requires that no hyperedge be monochromatic; i.e., at least two different colors are assigned to the vertices in every hyperedge consisting of two or more vertices.

An *edge coloring* of a hypergraph $H = (V, E)$ is a mapping $\phi : E \to \{1, 2, \ldots, p\}$ such that hyperedges e_1 and e_2 in E satisfy $\phi(e_1) \neq \phi(e_2)$ whenever $e_1 \cap e_2 \neq \emptyset$.

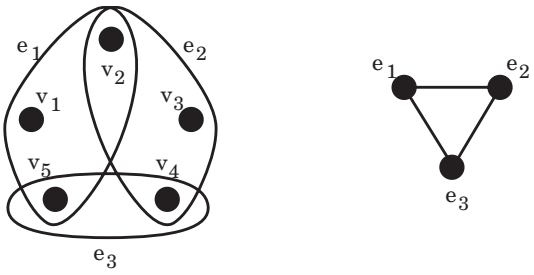

Fig. 9.1 *A hypergraph and its line graph.*

The strong (weak) vertex coloring problem of a hypergraph asks for a strong (weak) vertex coloring with the fewest colors. Similarly, the edge coloring problem of a hypergraph asks for an edge coloring with the fewest colors.

For more information on the concepts introduced here and other notions in hypergraphs as well as pointers to relevant literature, see, e.g., the book by Berge [13] and the survey article by Duchet [38]. The terms strong and weak coloring in hypergraphs are used, for instance, in [83, 112].

9.3. Representing Matrices Using Hypergraphs. To achieve our goal of formulating Problem 3.3 using hypergraphs, we introduce two new ways in which the sparsity structure of a matrix can be represented.

Let A be an $m \times n$ matrix. Define the *column-oriented* hypergraph representation of A as the hypergraph $H_c(A)$, where $V(H_c(A)) = \{v_1, v_2, \ldots, v_n\}$ and each v_j corresponds to column a_j; and $E(H_c(A)) = \{e_1, e_2, \ldots, e_m\}$ with each hyperedge $e_i = \{v_j : a_{ij} \neq 0\}$ "corresponding" to row r_i. The quotation marks are used to reflect that a row is perceived as a set of columns at which it has nonzero entries. Similarly, define the *row-oriented* hypergraph representation of A as the hypergraph $H_r(A)$, where $V(H_r(A)) = \{v'_1, v'_2, \ldots, v'_m\}$ and each v'_i corresponds to row r_i; and $E(H_r(A)) = \{e'_1, e'_2, \ldots, e'_n\}$ with each $e'_j = \{v'_i : a_{ij} \neq 0\}$ "corresponding" to column a_j.

9.4. Structurally Orthogonal Partition and Hypergraph Coloring. Our main result in this section is Theorem 9.1, which summarizes the equivalence relationships that underlie the graph- and hypergraph-theoretic formulations of Problem 3.3.

THEOREM 9.1. *Let A be an $m \times n$ matrix. Let H_c and H_r be the column-oriented and row-oriented hypergraph representations of A, respectively. Let G_b and G_c be the bipartite and column intersection graph representations of A, respectively. Let ϕ be a mapping $S \to \{1, 2, \ldots, p\}$, where S is a context-dependent set of n elements and p is a positive integer satisfying $p \leq n$. Then the following statements are equivalent:*

1. *ϕ induces a structurally orthogonal column partition of A.*
2. *ϕ is a partial distance-2 coloring of G_b on the column vertex set.*
3. *ϕ is a distance-1 coloring of G_c.*
4. *ϕ is a strong vertex coloring of H_c.*
5. *ϕ is an edge coloring of H_r.*
6. *ϕ is a distance-1 coloring of the line graph $L(H_r)$ of the hypergraph H_r.*

Proof. The equivalences among statements 1, 2, and 3 have already been established in section 3. Here we show the equivalences among the remaining statements; the reader will find Figure 9.2 helpful in following the discussion.

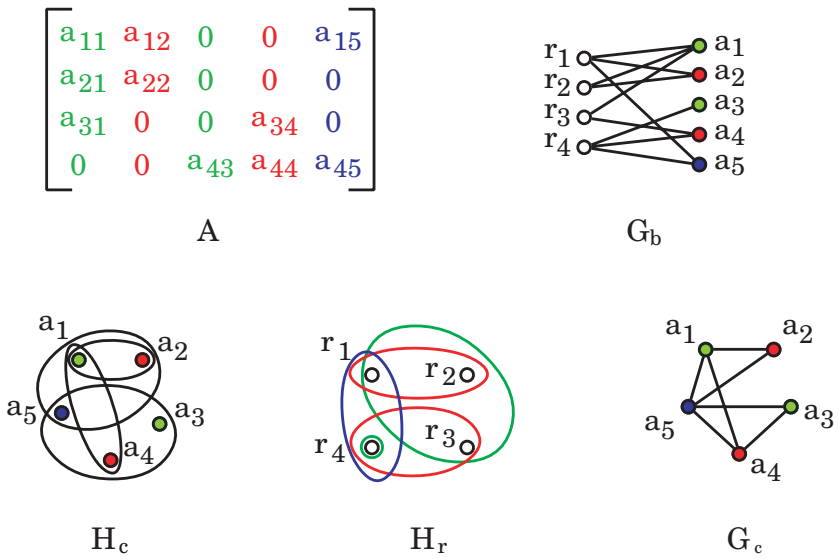

Fig. 9.2 *Structurally orthogonal partition and (hyper)graph colorings. In this illustration three colors are used: columns a_1 and a_3 are green, a_2 and a_4 are red, and a_5 is blue.*

First note the relationship between hypergraph H_c and column intersection graph G_c of A: the columns with a nonzero in row r_i form a clique in G_c, whereas they form a hyperedge in H_c. A similar observation can be made regarding H_r and the row intersection graph of A. With this observation it is obvious that a strong vertex coloring of H_c is equivalent to a distance-1 coloring of G_c. Next observe that the line graph $L(H_r)$ of H_r is isomorphic to the column intersection graph G_c. Thus statements 1, 2, 3, 4, and 6 are equivalent. Further, since the distance-1 neighbors of a vertex v from the set of column vertices of G_b constitute a hyperedge in H_r, it follows that statements 2 and 5 are equivalent, thus completing our proof. □

As we pointed out earlier in this section, the hypergraph formulation has a potential application in partial matrix computation for preconditioning. For such purposes, the idea is to relax the requirement on strong vertex coloring by allowing the reuse of colors within a hyperedge. Using such a weaker coloring, one may be able to compute a subset of the nonzero entries of a matrix and use the result as a preconditioner.

10. Other Matrix Estimation Methods. The methods for computing Jacobians and Hessians considered thus far in this paper rely on a number of underlying requirements.

First, each method is based on a unidirectional or a bidirectional *partition*; i.e., a set S of columns or rows is divided into disjoint subsets whose union is S. This requirement precludes the possibility of a column (or a row) in S belonging to more than one group, or to none of the groups. Second, a column or a row of a matrix is seen as an *atomic* entity; i.e., a column or a row is not divided into any smaller parts. Third, a partition is required to be structurally orthogonal, or symmetrically orthogonal, or substitutable. The last requirement implies that the system of equations defined by the partitions is either diagonal or triangular, and hence the unknowns can be obtained either directly or via substitution.

However, approaches where one or more of these underlying requirements are relaxed have also been suggested in the literature. In this section, we will discuss

a few examples of such approaches. For Hessian estimation, McCormick [97] gives a classification of direct methods, including those that do not necessarily rely on symmetrically orthogonal partitions.

10.1. Methods Based on Solving a Rectangular System. Newsam and Ramsdell [103] proposed a method for computing a Jacobian that relies on solving an overdetermined rectangular system of equations. The approach enables the determination of the nonzero entries of an $m \times n$ Jacobian matrix using ρ_{max} groups, where ρ_{max} is the maximum number of nonzeros in a row of the matrix. The method is optimal in terms of the number of groups used. However, it needs to solve n *least-squares* problems, and hence the system is potentially ill-conditioned.

Geitner, Utke, and Griewank [49] applied the Newsam and Ramsdell approach within the context of automatic differentiation.

Recently, Hossain and Steihaug [64] suggested an elimination scheme for computing a Jacobian. The scheme is based on successive merging of the columns of the *compressed* Jacobian, obtained from a structurally orthogonal partition. The nonzero entries are then determined by solving a banded system of equations. The latter requires an LU factorization. This approach also uses the optimal number of groups, the maximum number of nonzeros in a row of the Jacobian matrix.

10.2. Element and Variable Isolation. Newsam and Ramsdell [103] also suggested a method that generalizes structurally orthogonal column partitioning. They used the term *variable isolation* (VI) to refer to a method based on a structurally orthogonal column partition and *element isolation* (EI) to refer to their generalized method.

DEFINITION 10.1. *A nonzero element a_{ij} of a matrix A is* isolated *from nonzero element a_{pq}, whenever $a_{iq} = a_{pj} = 0$, or $j = q$ and $i \neq p$.*

The EI method partitions the nonzero entries of a matrix into groups consisting of pairwise isolated elements, with the objective of having the fewest groups. The nonzeros in a row of a matrix are, by definition, not isolated from each other; hence, the maximum number of nonzeros in a row is a lower bound for an optimal EI partition. An EI partition is used to determine a column grouping (in which a group may contain structurally nonorthogonal columns, and/or columns may belong to several groups) that enables the determination of all nonzero matrix entries via matrix-vector products. In particular, given a partition $I_1, I_2, \ldots, I_p$ of the nonzero entries of a matrix A into groups of isolated elements, a column grouping $C_1, C_2, \ldots, C_p$ is constructed as follows: $C_k = \{a_j : a_{ij} \in I_k \text{ for some } i\}$. In words, a column group C_k consists of the column indices of the nonzeros in a set I_k.

The middle column in Figure 10.1 illustrates two different partitions into isolated elements of a matrix whose nonzero structure is shown on the left. The upper part shows a partition into the four groups $\{a_{11}, a_{44}, a_{56}\}$, $\{a_{21}, a_{34}\}$, $\{a_{13}, a_{23}, a_{55}\}$, and $\{a_{26}, a_{35}, a_{42}\}$. The column grouping based on this element partition is then $\{a_1, a_4, a_6\}$, $\{a_1, a_4\}$, $\{a_3, a_5\}$, and $\{a_2, a_5, a_6\}$. The lower part shows an optimal partition consisting of three groups: $\{a_{11}, a_{21}, a_{35}, a_{42}, a_{55}\}$, $\{a_{13}, a_{23}, a_{34}, a_{44}\}$, and $\{a_{26}, a_{56}\}$. The column grouping corresponding to this EI partition is $\{a_1, a_2, a_5\}$, $\{a_3, a_4\}$, and $\{a_6\}$, which in this case coincides with a structurally orthogonal column partition.

Note that variable isolation in general implies element isolation but not vice versa. Thus an optimal EI method would use at most as many groups as an optimal VI method.

Newsam and Ramsdell formulated element isolation as a vertex coloring in a graph. The vertices in the graph are the nonzero entries of the matrix and an edge

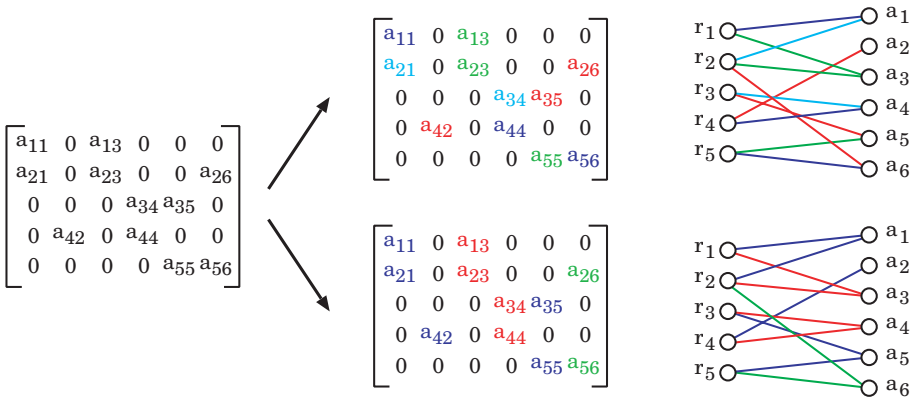

Fig. 10.1 *Two partitions of the nonzero entries of a matrix into groups of isolated elements. Each partition is also represented as a specialized edge coloring in the bipartite graph.*

between two entries exists whenever the latter are not isolated from each other. Such a coloring formulation is likely to be impractical for large problem instances as it deals with coloring a graph consisting of nnz vertices and $O((nnz)^2)$ edges, where nnz is the number of nonzero entries in the matrix.

10.3. Element Isolation and Edge Coloring. We note that the problem of partitioning the nonzero entries of a matrix into groups of isolated elements can in fact be modeled as a specialized *edge coloring* problem on a smaller graph.

DEFINITION 10.2. *Let $G_b = (V_1, V_2, E)$ be the bipartite graph of matrix A, where V_1 is the row vertex set and V_2 is the column vertex set. A mapping $\phi : E \rightarrow \{1, 2, \ldots, p\}$ is called an* EI edge coloring *if the following two conditions are satisfied:*

1. *At each row vertex v, edges having v as an endpoint are colored differently.*
2. *In every path (e_1, e_2, e_3) on three edges, $\phi(e_1) \neq \phi(e_3)$.*

From the definition of isolated elements, it is easy to see that an EI edge coloring in the bipartite graph $G_b(A)$ is equivalent to a partition of the nonzero entries of A into groups of isolated elements. The right column in Figure 10.1 shows the EI edge coloring representations of the partitions into isolated elements shown in the middle column.

10.4. Partitioning Segmented Columns. Hossain and Steihaug [61, 65] suggested a framework that has an EI and a VI method as its special cases. In particular, they suggest a technique for direct estimation of an $m \times n$ Jacobian in which the rows are first grouped into ℓ blocks that define "segmented" columns, and then the segments are partitioned into groups each of which consists of structurally *independent* segments. Two segments are said to be structurally independent in a manner analogous to two elements being isolated. The case $\ell = 1$ corresponds to a VI method and the case $\ell = m$ corresponds to an EI method. Hossain and Steihaug showed that, for some matrix structures, there exists an $\ell \neq 1$ that results in fewer groups compared with an approach based on $\ell = 1$. In a recent report [65], they characterized the nature of a partition that leads to an optimal number of groups. The optimal partition is also formulated as a vertex coloring in an associated graph. One of the main results in their work is that any direct determination of a Jacobian via matrix-vector products implies that the nonzero elements in each product are isolated elements.

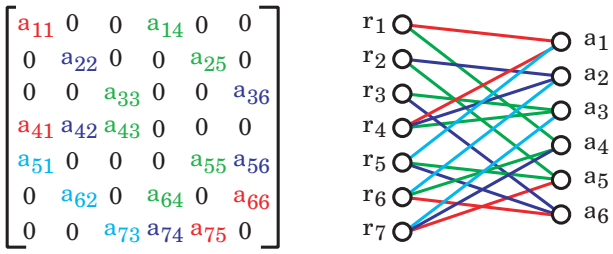

Fig. 10.2 *Eisenstat's example of a matrix where the nonzeros can be directly determined using fewer groups than are obtained from an optimal structurally orthogonal column partition.*

10.5. Eisenstat's Example. Eisenstat's (counter)example is cited often in the literature as a case in point where a Jacobian can be directly determined using fewer groups than that obtained by an optimal structurally orthogonal partition.

The example in its general form is stated for an $(n+1) \times n$ matrix A, where n is an even integer satisfying $n \geq 6$, in the following manner. Matrix $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ where $A_1 = \begin{bmatrix} D_1 & D_2 \end{bmatrix}$, $A_2 = \begin{bmatrix} C & 0 \\ D_3 & B \end{bmatrix}$, and D_1, D_2, $D_3 \in R^{\frac{n}{2} \times \frac{n}{2}}$ are nonsingular diagonal matrices; every diagonal entry of $B \in R^{\frac{n}{2} \times \frac{n}{2}}$ is zero, while every off-diagonal entry is nonzero; $C \in R^{1 \times \frac{n}{2}}$ consists entirely of nonzeros; and $0 \in R^{1 \times \frac{n}{2}}$ is a zero vector. Eisenstat's example for the case $n = 6$ is shown in Figure 10.2.

The columns in Eisenstat's example are pairwise structurally nonorthogonal, and hence an optimal structurally orthogonal column partition would require n groups. However, as has been noted by several authors (e.g., [31, 61, 103]), Eisenstat's example can be computed directly using $n/2 + 2$ groups by separately evaluating submatrices A_1 and A_2. Clearly, the columns of A_1 can be partitioned into two structurally orthogonal groups with columns of D_1 in the first and columns of D_2 in the second. For each i, $i \leq n/2$, columns i and $n/2 + i$ of A_2 are structurally orthogonal, and hence by grouping two such columns together, a structurally orthogonal partition of A_2 consisting of $n/2$ groups is possible. Hence, the entries of matrix A can be obtained from $n/2 + 2$ matrix-vector products. An evaluation of A in this manner corresponds to using $\ell = 2$ in the segmented column approach of Hossain and Steihaug.

For the case where $n = 6$ the discussion above shows that Eisenstat's example can be computed directly using five, instead of six, matrix-vector products. Hossain and Steihaug [65] have recently shown that it can be computed using four groups and that four is the optimal number of groups required in a direct method. They considered the column grouping $\{a_3, a_4, a_5\}$, $\{a_2, a_4, a_6\}$, $\{a_1, a_5, a_6\}$, and $\{a_1, a_2, a_3\}$ which corresponds to the seed matrix

$$D = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

The matrix-vector product AD from which the entries of A can be directly recovered

is thus

$$AD = \begin{bmatrix} a_{14} & a_{14} & a_{11} & a_{11} \\ a_{25} & a_{22} & a_{25} & a_{22} \\ a_{33} & a_{36} & a_{36} & a_{33} \\ a_{43} & a_{42} & a_{41} & a_{41}+a_{42}+a_{43} \\ a_{55} & a_{56} & a_{51}+a_{55}+a_{56} & a_{51} \\ a_{64} & a_{62}+a_{64}+a_{66} & a_{66} & a_{62} \\ a_{73}+a_{74}+a_{75} & a_{74} & a_{75} & a_{73} \end{bmatrix}.$$

We note that the EI edge coloring (see the left part in Figure 10.2) in which the edges are partitioned into the four groups $\{a_{11}, a_{41}, a_{66}, a_{75}\}$, $\{a_{14}, a_{25}, a_{33}, a_{43}, a_{55}, a_{64}\}$, $\{a_{22}, a_{36}, a_{42}, a_{56}, a_{74}\}$, and $\{a_{51}, a_{62}, a_{73}\}$ corresponds to the seed matrix of Hossain and Steihaug.

11. Miscellaneous Topics. Some of the coloring problems addressed in this paper have been studied outside the context of derivative matrix computation from a purely graph-theoretic or algorithmic point of view. In this section, we briefly review part of the relevant literature with emphasis on work that is close to the subject of this paper.

We begin in section 11.1 by discussing examples of effective vertex *orderings* for a greedy distance-1 coloring heuristic. In section 11.2 we include a relatively detailed discussion of the *coloring number* of a graph, a concept also useful in contexts other than graph coloring. In section 11.3 we mention a few results on upper and lower bounds for the chromatic number of a graph. Finally, in section 11.4 we provide an overview of complexity and approximability results for some of the coloring problems addressed in this paper. For further information on some of the issues discussed in sections 11.2–11.4, we refer the reader to Jensen and Toft [71] and Toft [115].

Our discussion in sections 11.1 through 11.3 pertains to distance-1 coloring. In those sections, for brevity, we write "coloring" instead of "distance-1 coloring." In section 11.4, however, we will explicitly state the kind of coloring being discussed. Unlike previous sections of this paper, the current section assumes familiarity with more advanced graph-theoretic concepts; we will use several such concepts without defining them.

11.1. Ordering for Coloring. In 1972, Matula, Marble, and Isaacson [94] analyzed various practically effective greedy coloring heuristics. A decade later Coleman and Moré [30] used, among others, some of these heuristics to solve a Jacobian estimation problem. Manvel [92], Kubale [85], and very recently Kosowski and Manuszewski [80] have surveyed greedy coloring heuristics.

A key issue in a greedy coloring heuristic is the *order* in which the vertices are visited as the latter determines the number of colors used by the heuristic. Here we discuss a few examples of effective vertex orderings.

For much of the discussion in this section we consider the greedy coloring heuristic SEQ, short for "sequential," outlined in Algorithm 11.1.

Notation. We begin by collecting some notation to be used here and in section 11.2. For a graph $G = (V, E)$ and a given vertex ordering $v_1, v_2, \ldots, v_n$, let $G[V_i]$ denote the graph *induced* by the vertex set $V_i = \{v_1, \ldots, v_i\}$. Hence $G[V_n] \equiv G$. Let q_i be the number of colors required by SEQ to color $G[V_i]$; thus q_n is the number of colors required by SEQ to color the entire graph G. Let $d(v, G[V_i])$ be the number of vertices adjacent to vertex v in $G[V_i]$; thus $d(v, G[V_n]) \equiv d(v)$ is the degree of vertex v in G. Recall that $\Delta(G)$ and $\delta(G)$ denote the maximum and minimum degree in

ALGORITHM 11.1. Greedy (sequential) coloring.

```
procedure SEQ(G = (V, E))
    Let v_1, v_2, ..., v_n be a vertex ordering
    Assign v_1 color 1
    for i ← 2 to n do
        Assign v_i the smallest color not used by any of its neighbors
    end for
end procedure
```

G, respectively. When the graph under discussion is clear from the context, we may simply use Δ and δ.

Theorems 11.1 and 11.3 below give upper bounds on the number of colors used by SEQ that reflect the importance of the chosen ordering; we will shortly see orderings that minimize the upper bounds given in the two theorems.

THEOREM 11.1. *The number of colors used by* SEQ *satisfies the inequality*

$$q_n \leq \max_{1 \leq i \leq n} \{d(v_i, G[V_i])\} + 1.$$

Proof. In the ith step, $i > 1$, if the algorithm does not introduce a new color to color vertex v_i, then by definition of q_i, we have $q_i = q_{i-1}$. Otherwise, $q_i = q_{i-1} + 1$. In the latter case, $d(v_i, G[V_i])$ satisfies the inequality $q_{i-1} \leq d(v_i, G[V_i])$ since a new color was needed to color v_i. The theorem follows by induction on i. ∎

The following result is an immediate consequence of Theorem 11.1.

COROLLARY 11.2. *For any ordering,* SEQ *uses at most* $\Delta + 1$ *colors.*

The theorem below gives a weakening of the bound given in Theorem 11.1.

THEOREM 11.3. *The number of colors used by* SEQ *satisfies the inequality*

$$q_n \leq \max_{1 \leq i \leq n} \min\{d(v_i), i - 1\} + 1.$$

Proof. Consider the bound in Theorem 11.1. Clearly $d(v_i, G[V_i]) \leq d(v_i, G) \equiv d(v_i)$. Also, since there are exactly $i - 1$ vertices in $G[V_i]$ other than v_i itself, the statement $d(v_i, G[V_i]) \leq i - 1$ holds—hence the bound in this theorem. ∎

Notice that a nonincreasing ordering of the vertices with respect to their degrees in the input graph G minimizes the upper bound in Theorem 11.3. This ordering, known as *largest first* (LF) ordering, was suggested by Welsh and Powell [120]. In general, however, the number of colors used by SEQ using an LF ordering can be much larger than the optimal number of colors. An example of a bipartite (hence 2-colorable) graph with n vertices and maximum degree $n/2 - 1$ for which an LF ordering could require $n/2$ colors is given in [30]. The example, as a by-product, shows that the bound of Corollary 11.2 is tight.

An ordering that has been found to usually require fewer colors compared with an LF ordering is a *smallest last* (SL) ordering. In SL ordering the last vertex in the ordering, v_n, is chosen to be a vertex with the minimum degree in G. The order of the remaining vertices is defined backwards. Assume that the vertices $v_{i+1}, \ldots, v_n$ in the ordering have been determined. Then vertex v_i is chosen to be a vertex with the minimum degree in the subgraph induced by the set $V - \{v_{i+1}, \ldots, v_n\}$. At each step, while choosing a vertex of minimum degree, ties are broken arbitrarily. Note that although an SL ordering is *computed* backwards, v_n through v_1, the vertices are then colored in the order v_1 through v_n. In this, SL ordering differs from LF ordering.

ALGORITHM 11.2. Greedy coloring formulated as a while-loop.

```
procedure SEQ-W(G = (V, E))
    U ← V                                  ▷ U is the set of currently uncolored vertices
    i ← 1
    while U ≠ ∅ do
        Choose a vertex v_i from U using some criterion
        Assign v_i the smallest color not used by any of its neighbors
        U ← U − {v_i}
        i ← i + 1
    end while
end procedure
```

The SL ordering was proposed by Matula [93] and, as discussed in section 11.2, its relevance in other contexts was discovered independently by other researchers.

In the context of algorithm SEQ, an SL ordering minimizes the bound given in Theorem 11.1. In particular, in an SL ordering, for each i, $1 \leq i \leq n$,

$$d(v_i, G[V_i]) = \min_{v_j \in V_i} d(v_j, G[V_i]).$$

An SL ordering of the vertices in a graph can be obtained in time proportional to the number of edges in the graph.

In the literature other vertex orderings, such as *incidence degree* (ID) and *saturation degree* (SD) ordering, have also been suggested as heuristics for reducing the number of colors used by a greedy algorithm. The former ordering is due to Coleman and Moré [30] and the latter is due to Brélaz [22]. ID and SD orderings are similar in spirit to LF ordering; they differ in the specialized "degree" measure they use. Unlike LF, SL, and ID orderings, an SD ordering can be computed only as the coloring proceeds. In other words, an SD ordering cannot be precomputed to then color the vertices.

To appreciate the ideas in LF, ID, and SD orderings, consider the scheme SEQ-W outlined in Algorithm 11.2. At the ith step of SEQ-W that uses LF ordering, among the current set of uncolored vertices, a vertex having the maximum *degree* in the input graph is selected and colored. An ID ordering–based coloring improves on this by focusing on the already colored part of the input graph. In particular, the ith step of SEQ-W that uses ID ordering selects a vertex (from the current set of uncolored vertices) having the maximum *incidence degree*, the number of already colored neighbors. Note that the incidence degree of vertex v_i is precisely $d(v_i, G[V_i])$.

In a further improvement, the ith step of SEQ-W that uses SD ordering chooses a vertex having the maximum *saturation degree*, the number of distinctly colored neighbors. In each of the three orderings LF, ID, and SD, ties are broken arbitrarily. The notions saturation degree (sd), incidence degree (id), and usual degree (d) of a vertex v satisfy the inequalities $sd(v) \leq id(v) \leq d(v)$. Figure 11.1 illustrates this; in the figure vertex c has the values two, three, and four for sd, id, and d, respectively. Notice that a vertex with a larger "degree" is more constrained in the choice of colors than a vertex with a smaller "degree." Note also that among the various specialized degree measures, saturation degree is the measure that most accurately models the choice of colors available to a vertex. Thus, intuitively one could expect algorithm SEQ-W to require a nondecreasing number of colors while using SD, ID, and LF orderings. Experimental results of Coleman and Moré [30] on graphs that arise in

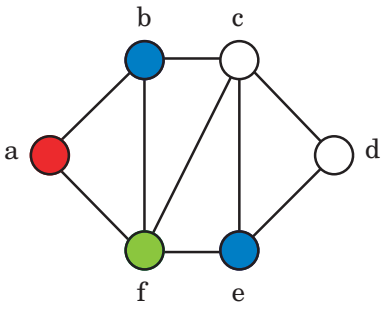

Fig. 11.1 *A partially colored graph illustrating the notions of saturation degree (sd), incidence degree (id), and "usual" degree (d). In this illustration, vertex a is red, vertices b and e are blue, and vertex f is green. Vertex c has sd = 2, id = 3, and d = 4.*

Table 11.1 *Summary of properties of SL, LF, ID, and SD ordering.*

Ordering	ith vertex satisfies
SL	$d(v_i, G[V_i]) = \min_{v_j \in V_i} d(v_j, G[V_i])$
LF	$d(v_i, G) = \max_{v_j \in (V - V_{i-1})} d(v_j, G)$
ID	$d(v_i, G[V_i]) = \max_{v_j \in (V - V_{i-1})} d(v_j, G[V_{i-1} \cup \{v_j\}])$
SD	$sd(v_i, G[V_i]) = \max_{v_j \in (V - V_{i-1})} sd(v_j, G[V_{i-1} \cup \{v_j\}])$

practice by and large support this intuition. In terms of time complexity, algorithm SEQ-W using LF or ID ordering can be implemented to run in $O(m)$ time, whereas an SD ordering–based coloring runs in $O(n^2)$ time [22, 30, 120]. Here n and m are the numbers of vertices and edges, respectively.

In Table 11.1 we give a succinct summary of the properties of SL, LF, ID, and SD orderings. The right column of the table shows a statement satisfied by the ith vertex in the respective ordering, for each i, $1 \le i \le n$; in the case where $i = 1$, the set V_{i-1} on the right column corresponds to the empty set. The quantity $sd(v, G[V_i])$ in the last row of the table shows the saturation degree of the vertex v in the graph $G[V_i]$. Notice that any vertex in G can be chosen to be v_1 in the case of ID and SD orderings.

Coleman and Moré [30] showed that algorithm SEQ-W that uses an ID ordering requires two colors on bipartite graphs and hence is optimal for this class of graphs. They also showed that this is not the case for SL ordering by giving an example of a bipartite graph on n vertices in which SEQ using an SL ordering could require $n/3+1$ colors.

It should be pointed out that there exists an ordering of the vertices in which SEQ would use the optimal number of colors. To see this, consider an optimal coloring $\phi : V \to \{1, 2, \ldots, p\}$ of a graph $G = (V, E)$. Let the vertices be ordered such that the sequence $\{\phi(v_i)\}$ is nondecreasing, i.e., $\phi(v_i) \le \phi(v_{i+1})$ for $1 \le i \le n-1$. Note that in such an ordering, vertices of each color class are listed consecutively. It is then easy to see that if the vertices were to be colored *afresh* in the order just described, algorithm SEQ would produce a coloring with an optimal number of colors. The problem while

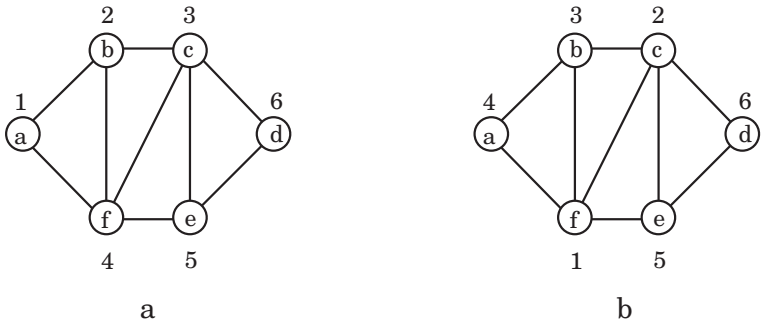

Fig. 11.2 *Two vertex orderings of a graph with different values for maximum* back-degree*; the maximum back-degree in* (a) *is three and in* (b) *is two (which is optimal).*

using SEQ, of course, is finding an ordering among the $n!$ possibilities that would produce an optimal coloring. The fact that there exists an ordering in which SEQ would use the optimal number of colors is unlike a number of other problems in optimization in which a greedy algorithm provided with the "right ordering" might still fail to yield an optimal solution.

11.2. The Coloring Number. In a given ordering of the vertices of a graph, let the *back-degree* of a vertex v_i be the number of adjacent vertices to v_i that precede v_i in the ordering. Using the notation introduced in section 11.1, the back-degree of vertex v_i is the same as $d(v_i, G[V_i])$, a quantity that is also referred to as the incidence degree of vertex v_i. For a given graph $G = (V, E)$, let $B_\pi(G)$ denote the maximum back-degree in vertex ordering $\pi : V \leftrightarrow \{1, 2, \ldots, n\}$. Clearly, different vertex orderings may result in different maximum back-degrees. As an illustration, Figure 11.2 shows two different vertex orderings in a graph. The maximum back-degree in the ordering shown in part (a) is three and the corresponding value in part (b) is two.

The quantity $\min_\pi\{B_\pi(G)\}$, where the minimum is taken over the $n!$ possible vertex orderings, is of special interest. Let this quantity be denoted by $B^*(G)$. For the graph shown in Figure 11.2, the ordering shown in part (b) is optimal, i.e., $B^*(G) = 2$.

Consider the coloring algorithm SEQ in which the vertex ordering used is such that $B^*(G)$ is attained. With such an ordering, in each step i of SEQ, vertex v_i has at most $B^*(G)$ already colored neighbors and hence can be colored using one value from the set $\{1, 2, \ldots, B^*(G) + 1\}$. Thus $B^*(G) + 1$ colors suffice to color G. Motivated by this fact, the quantity $B^*(G) + 1$ is known in the literature as the *coloring number* of G and is denoted by $col(G)$. The name coloring number was first used by Erdös and Hajnal [40]. As we will shortly see, the graph parameter $B^*(G)$ is also of interest in contexts other than coloring.

The coloring number may appear to be an intractable graph parameter as the minimum is evaluated over $n!$ possible orderings. However, this was proven to be not the case independently by Finck and Sachs [42] and Matula [93]. Theorem 11.4 reflects this fact. It states that the number of colors required by algorithm SEQ on a graph G with an SL ordering is the coloring number of G. Recall that an SL ordering can be computed in linear time in the number of edges in a graph.

THEOREM 11.4. *Every graph G satisfies*

$$col(G) = B_{SL}(G) + 1.$$

Proof. By definition, $col(G) = B^*(G) + 1$. Clearly $B^*(G) \leq B_{SL}(G)$. Hence $col(G) \leq B_{SL}(G)+1$. We now show that the inequality holds in the opposite direction as well, thus implying equality.

Consider an SL ordering $S = v_1, v_2, \ldots, v_n$ of the vertices of G. Clearly, for every $i < n$, $col(G) \equiv col(G[V_n]) \geq col(G[V_i])$; furthermore, $col(G[V_i]) \geq \delta(G[V_i]) + 1$, since the back-degree of vertex v_i in the subsequence $v_1, v_2, \ldots, v_i$ of S is its ordinary degree in $G[V_i]$, which in turn is equal to $\delta(G[V_i])$ since S is an SL ordering. Hence $col(G) \geq \max_i\{\delta(G[V_i])\} + 1$. The quantity on the right side of the inequality in the preceding statement is precisely $B_{SL}(G) + 1$. Thus we have $col(G) \geq B_{SL}(G) + 1$, which completes the proof. □

Matula [93] and Szekeres and Wilf [113], independently, have related the coloring number of a graph to yet another seemingly intractable graph parameter. The parameter we are referring to is the maximum minimum degree in an induced subgraph of a graph, where the maximum is taken over all possible induced subgraphs. In a given graph $G = (V, E)$, since there are 2^n possible subsets of V, computing the maximum minimum degree in an induced subgraph of G appears to be intractable. Theorem 11.5 shows this is not the case. Let $H \subseteq G$ denote a nonempty induced subgraph of G.

THEOREM 11.5. *Every graph G satisfies*

$$col(G) = \max_{H \subseteq G}\{\delta(H)\} + 1.$$

Proof. Let $k = \max_{H \subseteq G}\{\delta(H)\}$. We claim that the graph $G = (V, E)$ itself has a vertex of degree at most k, for otherwise the minimum degree in G, $\delta(G)$, is strictly greater than k, contradicting the definition of k. Let v_n be a vertex with degree $\leq k$ in the graph G and let $H_{n-1} = G[V - \{v_n\}]$. By assumption, H_{n-1} has a vertex of degree at most k. Let v_{n-1} be one of them and let $H_{n-2} = G[V - \{v_n, v_{n-1}\}]$. Continue in this manner to obtain the ordering $v_1, v_2, \ldots, v_n$ of the vertices of G. It is then clear that in such an ordering the maximum back-degree of a vertex is at most k. Hence $col(G) \leq k + 1$.

We now prove that the inequality holds in the other direction as well. For any $H \subseteq G$, clearly $col(G) \geq col(H)$ and $col(H) \geq \delta(H) + 1$, since the back-degree of the last vertex in any ordering of the vertices of H is just its ordinary degree in H, which is at least $\delta(H)$. Thus $col(G) \geq k + 1$. The theorem follows since $col(G)$ is shown to be both less than or equal to and greater than or equal to the quantity $\max_{H \subseteq G}\{\delta(H)\} + 1$. □

Clearly, the coloring number of a graph is a tighter upper bound on its chromatic number than the bound $\Delta + 1$; i.e., for any graph G the inequalities $\chi(G) \leq col(G) \leq \Delta(G) + 1$ hold. Moreover, $col(G) = \Delta(G) + 1$ if and only if G has a $\Delta(G)$-regular connected component.

A graph G for which $col(G) \leq k+1$ has also been called *k-degenerate* by Lick and White [88]. (A graph is k-degenerate if every induced subgraph has minimum degree at most k.) A related concept has been studied in the social and biological network literature as the *k-core* of a graph [21, 105]. The k-core of a graph is a maximal induced subgraph in which every vertex has at least k neighbors in the subgraph. A k-core of a graph can be computed by a linear-time algorithm that visits the vertices in the reverse of an SL ordering. A paper by Ramadan, Tarafdar, and Pothen [109] extends the definition of the k-core to a hypergraph and discusses the k-cores of protein interaction graphs and protein complex hypergraphs. The coloring number of a graph is also closely related to the *arboricity* of the graph [37].

11.3. Bounds for the Chromatic Number. Since computing the chromatic number $\chi(G)$ of a general graph G is NP-hard, we consider bounding it with more tractable graph parameters.

Upper Bounds. As we have already seen, the coloring number $col(G)$ is a linear-time computable upper bound for the chromatic number $\chi(G)$ of a graph G. We have also seen that $\Delta(G)+1$ is another easy to compute (but weaker) upper bound for $\chi(G)$. If G is a complete graph or an odd cycle, then in fact $\chi(G) = \Delta(G)+1$. In all other cases, Brooks's theorem tells us that the general bound $\Delta(G)+1$ can be improved by 1.

THEOREM 11.6 (Brooks's theorem). *Let G be a connected graph. If G is neither complete nor an odd cycle, then $\chi(G) \leq \Delta(G)$.*

For a proof of Brooks's theorem, see, e.g., Diestel [37].

Lower Bounds. The size of the largest clique in G, $\omega(G)$, is one obvious lower bound for $\chi(G)$. Another, less obvious, lower bound comes from the fact that a graph coloring is a vertex partition into independent sets. If G does not contain an independent set of size $s+1$, then in every coloring of G at most s vertices get the same color. Hence, $\chi(G) \geq |V|/\alpha(G)$, where $\alpha(G)$ is the size of the largest independent set in $G = (V, E)$. Combining the two, one gets the following improved lower bound:

$$\chi(G) \geq \max\{\omega(G), |V|/\alpha(G)\}.$$

For many graphs the bound just given is a weak one. Moreover, both $\omega(G)$ and $\alpha(G)$ are NP-hard to determine. By considering a more general class of subgraphs than cliques, Hajós [57] obtained a further improved lower bound (whose computation is not known to be polynomial). In particular, Hajós showed that $\chi(G) \geq k$ if and only if G has a *k-constructible* subgraph. For further discussion on Hajós's and other lower bounds, see, e.g., Diestel [37] or Toft [115].

11.4. Theoretical Results on Coloring Problems. In this last subsection, we provide an overview of various results in the graph theory literature on some of the coloring problems addressed in this paper. Our main objective here is to point the interested reader to relevant references.

Each of the NP-hardness results stated in the next paragraph has been mentioned elsewhere in this paper; here the results are presented together for convenience.

Lin and Skiena [89] proved that the distance-k graph coloring problem is NP-hard for every fixed integer $k \geq 1$. The proof relies on the equivalence between distance-k coloring of G and distance-1 coloring of G^k. Coleman and Moré [31] showed that the star coloring problem is NP-hard even if the graph is bipartite. Coleman and Verma [32] showed that the problem of finding a star bicoloring using the fewest colors is also NP-hard. Further, the acyclic coloring problem was proven to be NP-hard by Coleman and Cai [27].

For a general graph on n vertices, the first approximation algorithm for distance-1 coloring, with an approximation ratio of $O(n\frac{1}{\log n})$, was obtained by Johnson [72]. This was later improved to $O(n(\frac{\log\log n}{\log n})^2)$ by Wigderson [121] and to $O(n(\frac{\log\log n}{\log n})^3)$ by Berger and Rompel [14]. The current best known approximation ratio is $O(n\frac{(\log\log n)^2}{(\log n)^3})$, a result due to Halldórsson [59]. Based on Wigderson's algorithm, Karger, Motwani, and Sudan [75] considered approximate graph coloring using *semidefinite programming* techniques. On the negative side, Bellare, Goldreich, and Sudan [12] showed that distance-1 coloring is not approximable within $O(n^{1/7-\epsilon})$ for any $\epsilon > 0$, unless P = NP.

A planar graph can be distance-1 colored using four colors. This fact, best known as the *four color theorem*, is perhaps one of the most famous results in graph theory. After challenging mathematicians for nearly a century, the theorem was proved by Appel and Haken in 1976 [7, 10]. However, due to its length, its extensive use of computer verification, and its omission of details, the proof of Appel and Haken has been subject to criticism. In response, the authors published a revised version of the proof a decade later [8, 9]. Jensen and Toft [71] cited a 1994 manuscript of Robertson, Sanders, Seymour, and Thomas that presents a new, highly simplified proof of the four color theorem. For further information on the four color theorem, see the recent books by Fritsch and Fritsch [43] and Wilson [122].

Since every planar graph has a vertex of degree at most five, one can give a linear-time algorithm for finding a distance-1 coloring of a planar graph that uses at most six colors: run algorithm SEQ with an SL ordering. A linear-time algorithm that finds a distance-1 coloring of a planar graph using five colors was presented by Matula, Shiloach, and Tarjan [95, 96] and independently by Chiba, Nishizeki, and Saito [25]. Also, a linear-time algorithm that uses five colors is implicit in a recent proof of the theorem "every planar graph is five colorable" given by Thomassen [114]. Deciding whether a planar graph of maximum degree four can be distance-1 colored using three colors is NP-complete [71].

Wegner [119] studied distance-k ($k \geq 2$) chromatic numbers of planar graphs having maximum degree at most three and proved that for $k = 2$ eight colors suffice, whereas seven is the best possible. Recently, Agnarsson and Halldórsson [3] studied distance-k coloring of planar graphs. They showed that a planar graph having a sufficiently large maximum degree Δ can be distance-2 colored using at most $\lceil 9\Delta/5 \rceil$ colors. The authors also showed that for a fixed integer $k \geq 1$, the distance-k chromatic number of a planar graph is $O(\Delta^{\lfloor k/2 \rfloor})$. Agnarsson, Greenlaw, and Halldórsson [2] studied distance-k coloring of chordal graphs and showed that for even integers k, the problem is not approximable within a factor of $|V|^{1/2-\epsilon}$ for any $\epsilon > 0$.

Acyclic colorings were first defined and studied by Grünbaum [56]. Grünbaum asked whether the acyclic chromatic number $\chi_a(G)$ of a graph G satisfies the inequality $\chi_a(G) \leq \Delta + 1$. A negative answer to this question was given by Erdös, who proved probabilistically the existence of graphs where $\chi_a(G) \geq \Delta^{4/3-\epsilon}$ [71]. Alon, McDiarmid, and Reed [6] showed (again by probabilistic methods) that there exists a constant $c_1 > 0$ such that $\chi_a(G) \leq c_1 \cdot \Delta^{4/3}$. The same authors also showed that there is a constant $c_2 > 0$ such that there exist graphs with $\chi_a(G) > c_2 \cdot \Delta^{4/3} \cdot (\log \Delta)^{-1/3}$ for infinitely many Δ. There are a few studies concerning $\chi_a(G)$ for specific values of Δ. It is clear that $\chi_a(G) \leq 2$ if and only if G is a forest. Grünbaum [56] proved that $\chi_a(G) \leq 4$ for $\Delta = 3$, and $\chi_a(G) \leq 6$ for $\Delta = 4$. The latter bound was reduced from six to five by Burstein [23]. Kostochka [82] proved that it is NP-complete to decide whether a given graph G satisfies $\chi_a(G) \leq 3$.

Grünbaum [56] also proved that every planar graph admits an acyclic coloring using at most nine colors. After a series of improvements [4, 81, 98] the number was finally reduced to five by Borodin [19]. Borodin's proof is similar to the proof of the four color theorem in that there are 450 reducible configurations that need to be checked.

Star coloring has been recently studied by Albertson et al. [5] who show that every acyclic coloring that uses q colors can be refined to a star coloring with at most $2q^2 - q$ colors. They also prove that planar graphs have star colorings with at most 20 colors and exhibit an example in which 10 colors are required. This is an improvement on the upper bounds 80 and 30, obtained earlier by Fertin, Raspaud, and Reed [41] and

Neśetŕil and Ossan de Mendez [102], respectively. The problem of deciding whether a planar bipartite graph has a star coloring with at most three colors was shown to be NP-complete in [5].

12. Conclusion.

12.1. Summary. We have studied the efficient computation of sparse Jacobian and Hessian matrices using finite difference and automatic differentiation techniques. In this context, we considered a variety of matrix partitioning problems that can be classified along four axes. The first axis corresponds to whether the matrix to be computed is nonsymmetric (Jacobian) or symmetric (Hessian). The second axis relates to whether the matrix partition used is unidirectional (involving either columns or rows) or bidirectional (involving both columns and rows). The third axis corresponds to whether the matrix entries are determined directly (by solving a diagonal system of equations) or via substitution (by solving a triangular system of equations). The last axis shows whether all the entries of the matrix need to be computed or just a subset of them suffices for preconditioning purposes. Out of the sixteen possibilities this scenario suggests, ten cases correspond to meaningful problems within our framework, and eight of them are addressed in this paper.

Building upon several pioneering papers that have been published in the last twenty years, we have developed a unifying graph-theoretic framework for studying these matrix partitioning problems.

Our problem formulations and algorithms are based on robust and space-efficient graph representations of the sparsity structure of matrices: The Jacobian is represented by its bipartite graph, and the Hessian by its adjacency graph.

We showed that the distance-2 graph coloring problem is a generic model for the various matrix partitioning problems. Our approach provides fresh insight into the matrix partitioning problems, leading to several simple and effective heuristic algorithms. Among others, we developed fast star coloring heuristics, sketched a new heuristic algorithm for the acyclic coloring problem, and suggested a scheme for solving bicoloring problems.

We have also shown a hypergraph coloring formulation for the unidirectional Jacobian computation problem. This alternative formulation may be helpful in partial matrix computation for preconditioning purposes.

In the case of unidirectional Jacobian computation via a direct method, we showed experimentally the advantages offered by the new partial distance-2 coloring formulation as compared to the previously known distance-1 coloring formulation. In computing the Hessian using a direct method, we demonstrated a time/quality trade-off between two star coloring algorithms.

In general, most of the algorithms in the literature for solving the coloring problems considered in this paper rely on first transforming the input graph $G = (V, E)$ to some denser graph $G' = (V, E')$, $E' \supseteq E$, such that a distance-1 coloring of G' is equivalent to the particular coloring problem on G [27, 31, 32]. In contrast, the algorithms proposed in this paper solve the particular coloring problem directly on G. The main advantages offered by our approach include: smaller storage space requirement, the possibility to mix-and-match methods, and ease of developing more efficient algorithms and flexible software.

Our primary emphasis has been on computational methods that treat a column or a row of a matrix as an indivisible unit and that *partition* the columns or rows (or both). However, we have briefly reviewed methods that do not fit this framework. We formulated one of these methods, element isolation, as a specialized *edge coloring* in the bipartite graph of the Jacobian.

We have also included a detailed discussion of effective vertex orderings for greedy coloring, a discussion of the *coloring number* of a graph and its significance in various contexts, and a brief review of relevant theoretical results from the graph coloring literature.

12.2. Recent Development. During the time this paper was under review or in press, we have made progress on two fronts. First, we have implemented the acyclic coloring algorithm sketched in section 6.1.3 and developed a new algorithm for the star coloring problem; both algorithms are based on the paradigm of maintaining two-colored subgraphs. A paper that discusses these algorithms, and their implementation and performance has been submitted elsewhere [48]. Our implementations of the new acyclic and star coloring algorithms are currently being incorporated into the automatic differentiation software ADOL-C [118]. Second, we have implemented the various vertex orderings discussed in section 11.1, each tailored for the distance-k ($k = 1, 2$) coloring problem. In a future paper, we plan to report experimental results demonstrating the impact of the various ordering techniques when sequential coloring algorithms are applied to large-scale problems. All software we develop will be made publicly available.

12.3. Further Work. This work can be followed up along several directions, some of which we list below. We hope that these problems will attract the attention of our readers.

1. The new algorithms for star bicoloring (section 5.4) and acyclic bicoloring (section 6.2.3) suggested in this paper need to be implemented.
2. New algorithms need to be designed and implemented for the newly considered partial matrix computation problems (section 8).
3. One of the motivations for the current study has been the need to develop parallel algorithms for solving partitioning problems in large-scale optimization contexts. In previous work [45, 47] we developed shared-memory parallel algorithms for the distance-2 and star coloring problems. Some of the ideas in these algorithms have recently been extended to design and implement distributed memory parallel algorithms for the distance-1 and the distance-2 coloring problems [18, 20]. Based on these early experiences and the current study, scalable distributed-memory parallel algorithms for the various coloring problems need to be designed and implemented.
4. In the case of bidirectional partitioning problems, based on the known relationship to graph bicoloring, we have argued that finding a "small"-size vertex cover in a preprocessing step contributes to making the overall computation more efficient (section 5.4). Loosely speaking, the desired vertex cover has to favor high-degree vertices from both vertex sets of the bipartite graph. A precise characterization of the "optimum" vertex cover required is an issue that we are currently studying.
5. The maximum number of nonzeros in a row of a Jacobian is a lower bound on the number of colors required to distance-2 color the column vertices of the associated bipartite graph $G_b = (V_1, V_2, E)$. A tighter lower bound is the size of a maximal clique in the graph $G_b^2[V_2]$ (section 3.5.2). How can one find such a clique and its size quickly?
6. What nontrivial lower bounds can be given for star coloring (section 4)? for star bicoloring (section 5)? for acyclic coloring (section 6.1)? for acyclic bicoloring (section 6.2)?
7. Are there special graph classes for which these problems are tractable or have good upper bounds?

8. This paper dealt with coloring formulations of Jacobian and Hessian computation problems where the sparsity pattern is not necessarily *regular*. In cases where the pattern is regular, such as Jacobians or Hessians that arise in discretization of structured grids, the resulting coloring problems can be solved optimally in polynomial time (see, e.g., the work of Goldfarb and Toint [52]). A closer look at coloring problems for such structured graphs is a worthwhile issue.
9. For the variant of edge coloring discussed in section 10.3, the maximum degree in the set of row vertices of the bipartite graph is a lower bound for the fewest colors needed. What upper bound can be given? What exact/approximation algorithm can be designed?
10. Star coloring a planar graph requires at most twenty colors, and an example is known for which ten colors are needed (section 11.4). Reducing the gap between the lower bound and the upper bound is an open problem.

Graph coloring for efficiently computing Jacobians and Hessians represents one of the earliest applications of combinatorial methods in numerical optimization. This is but one of the problems in the emerging research area of *combinatorial scientific computing*, an area where combinatorial models and algorithms are used to solve problems in scientific computing. Our broader hope is that this article provides an impetus for the formulation and solution of many more combinatorial problems in scientific computing.

Acknowledgments. This work was begun when Assefaw Gebremedhin visited Alex Pothen at Old Dominion University for a semester during his Ph.D. study at the University of Bergen, Norway; Fredrik Manne was Assefaw's advisor. Paul Hovland and Rob Bisseling, who served on Assefaw's thesis examination committee, read earlier versions of some of the sections of this paper and suggested several improvements and new problems to consider. Rob Bisseling has also read a recent version of this paper and shared his constructive remarks with us. Trond Steihaug at the University of Bergen has given us many helpful comments. Tom Coleman encouraged our work on graph coloring for optimization, even when others had felt that "it has all been done already." Paul Hovland and Lois McInnes have been enthusiastic about using this work in their software for AD and optimization. Our work on partial estimation problems were stimulated by conversations with Bill Spotz at Sandia National Labs. Bruce Hendrickson, Erik Boman, Robert Preis (currently at the University of Paderborn, Germany), and other members of the Discrete Algorithms Group at Sandia National Labs have discussed these results with us during Alex Pothen's sabbatical visits to Sandia. Dinesh Kaushik gave us some insight on the runtime of a finite difference code for computing the numerical values of the entries of a large, sparse Jacobian matrix. Randy LeVeque and Michael Overton have read an earlier version of this paper and given us helpful feedback and encouragement. The anonymous referees have provided helpful comments on presentation. This paper has been shaped and improved by our interactions with these supportive colleagues, and our thanks go to all of them.

REFERENCES

[1] A. Aggarwal, A. Bar-Noy, D. Coppersmith, R. Ramaswami, B. Schieber, and M. Sudan, *Efficient routing and scheduling algorithms for optical networks*, in Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 1994, pp. 412–423.

[2] G. Agnarsson, R. Greenlaw, and M. M. Halldórsson, *On powers of chordal graphs and their colorings*, Congr. Numer., 100 (2000), pp. 41–65.
[3] G. Agnarsson and M. M. Halldórsson, *Coloring powers of planar graphs*, SIAM J. Discrete Math., 16 (2003), pp. 651–662.
[4] M. O. Albertson and D. M. Berman, *The acyclic chromatic number*, Congr. Numer., 17 (1976), pp. 51–69.
[5] M. O. Albertson, G. G. Chappell, H. A. Kierstead, A. Kündgen, and R. Ramamurthi, *Coloring with no* 2*-colored* P_4*'s*, Electron. J. Combin., 11 (2004), article R26.
[6] N. Alon, C. McDiarmid, and B. Reed, *Acyclic coloring of graphs*, Random Structures Algorithms, 2 (1991), pp. 277–288.
[7] K. Appel and W. Haken, *Every planar map is four colorable, Part* I: *Discharging*, Illinois J. Math., 21 (1977), pp. 429–490.
[8] K. Appel and W. Haken, *The four color proof suffices*, Math. Intell., 8 (1986), pp. 10–20.
[9] K. Appel and W. Haken, *Every Planar Map Is Four Colorable*, AMS, Providence, RI, 1989.
[10] K. Appel, W. Haken, and J. Koch, *Every planar map is four colorable, Part* II: *Reducibility*, Illinois J. Math., 21 (1977), pp. 491–567.
[11] O. Axelsson and U. Nävert, *On a graphical package for nonlinear partial differential equation problems*, in Proceedings of the IFIP Congress 77, B. Gilchrist, ed., Information Processing, North-Holland, Amsterdam, 1977, pp. 103–108.
[12] M. Bellare, O. Goldreich, and M. Sudan, *Free bits, PCPs and nonapproximability—towards tight results*, SIAM J. Comput., 27 (1998), pp. 804–915.
[13] C. Berge, *Hypergraphs*, North-Holland, Amsterdam, 1989.
[14] B. Berger and J. Rompel, *A better performance guarantee for approximate graph coloring*, Algorithmica, 5 (1990), pp. 459–466.
[15] J. Bialogrodzki, *Path coloring and routing in graphs*, in Graph Colorings, M. Kubale, ed., Contemp. Math. 352, AMS, Providence, RI, 2004, pp. 139–152.
[16] C. Bischof, A. Carle, P. Khademi, and A. Mauer, *ADIFOR* 2.0: *Automatic differentiation of Fortran* 77 *programs*, IEEE Comput. Sci. Engrg., 3 (1996), pp. 18–32.
[17] C. Bischof, L. Roh, and A. Mauer, *ADIC: An extensible automatic differentiation tool for ANSI-C*, Software—Practice and Experience, 27 (1997), pp. 1427–1456.
[18] E. Boman, D. Bozdağ, U. Catalyurek, A. H. Gebremedhin, and F. Manne, *A scalable parallel graph coloring algorithm for distributed memory computers*, in Proceedings of Euro-Par 2005 Parallel Processing, J. Cunha and P. Medeiros, eds., Lecture Notes in Comput. Sci. 3648, Springer-Verlag, Berlin, 2005, pp. 241–251.
[19] O. V. Borodin, *On acyclic colorings of planar graphs*, Discrete Math., 25 (1979), pp. 211–236.
[20] D. Bozdağ, U. Catalyurek, A. H. Gebremedhin, F. Manne, E. Boman, and F. Özgüner, *A parallel distance-*2 *graph coloring algorithm for distributed memory computers*, in Proceedings of HPCC-05, the 2005 International Conference on High Performance Computing and Communications, L.T. Yang et al., eds., Lecture Notes in Comput. Sci. 3726, Springer-Verlag, Berlin, 2005, pp. 796–806.
[21] B. J. Breitkreutz, C. Stark, and M. Tyers, *Osprey: A network visualization system*, Genome Bio., 4 (2003), article R22.
[22] D. Brélaz, *New methods to color the vertices of a graph*, Comm. ACM, 22 (1979), pp. 251–256.
[23] M. I. Burstein, *Every* 4*-valent graph has an acyclic* 5*-coloring*, Soobsc. Akad. Nauk Gruzin. SSR, 93 (1979), pp. 21–24 (in Russian, with Georgian and English summaries).
[24] G. J. Chaitin, M. Auslander, A. K. Chandra, J. Cocke, M. E. Hopkins, and P. Markstein, *Register allocation via coloring*, Comput. Lang., 6 (1981), pp. 47–57.
[25] N. Chiba, T. Nishizeki, and N. Saito, *A linear* 5*-coloring algorithm for planar graphs*, J. Algorithms, 2 (1981), pp. 317–327.
[26] T. F. Coleman, *Large Sparse Numerical Optimization*, Lecture Notes in Comput. Sci. 165, Springer-Verlag, New York, 1984.
[27] T. F. Coleman and J.-Y. Cai, *The cyclic coloring problem and estimation of sparse Hessian matrices*, SIAM J. Algebraic Discrete Methods, 7 (1986), pp. 221–235.
[28] T. F. Coleman, B. Garbow, and J. J. Moré, *Software for estimating sparse Jacobian matrices*, ACM Trans. Math. Software, 10 (1984), pp. 329–347.
[29] T. F. Coleman, B. Garbow, and J. J. Moré, *Software for estimating sparse Hessian matrices*, ACM Trans. Math. Software, 11 (1985), pp. 363–377.
[30] T. F. Coleman and J. J. Moré, *Estimation of sparse Jacobian matrices and graph coloring problems*, SIAM J. Numer. Anal., 20 (1983), pp. 187–209.
[31] T. F. Coleman and J. J. Moré, *Estimation of sparse Hessian matrices and graph coloring problems*, Math. Program., 28 (1984), pp. 243–270.

[32] T. F. Coleman and A. Verma, *The efficient computation of sparse Jacobian matrices using automatic differentiation*, SIAM J. Sci. Comput., 19 (1998), pp. 1210–1233.
[33] G. Corliss, C. Faure, A. Griewank, L. Hascoët, and U. Naumann, *Automatic Differentiation of Algorithms: From Simulation to Optimization*, Springer-Verlag, New York, 2002.
[34] J. Culberson, *Graph coloring page*, http://www.cs.ualberta.ca/~joe/Coloring/index.html.
[35] A. R. Curtis, M. J. D. Powell, and J. K. Reid, *On the estimation of sparse Jacobian matrices*, J. Inst. Math. Appl., 13 (1974), pp. 117–119.
[36] T. Davis, *University of Florida Sparse Matrix Collection*, NA Digest, 97 (1997); http://www.cise.ufl.edu/research/sparse/matrices.
[37] R. Diestel, *Graph Theory*, 2nd ed., Springer-Verlag, New York, 2000.
[38] P. Duchet, *Hypergraphs*, in Handbook of Combinatorics, Volume I, R. L. Graham, M. Grötschel, and L. Lovász, eds., MIT Press, Cambridge, MA, 1996, pp. 381–432.
[39] E. Elmroth, F. Gustavson, I. Jonsson, and B. Kågström, *Recursive blocked algorithms and hybrid data structures for dense matrix library software*, SIAM Rev., 46 (2004), pp. 3–45.
[40] P. Erdös and A. Hajnal, *On chromatic number of graphs and set systems*, Acta. Math. Acad. Sci. Hungar., 17 (1966), pp. 61–99.
[41] G. Fertin, A. Raspaud, and B. Reed, *On star coloring of graphs*, in Graph-Theoretic Concepts in Computer Science, 27th International Workshop, Lecture Notes in Comput. Sci. 2204, Springer-Verlag, Berlin, 2001, pp. 140–153.
[42] H. J. Finck and H. Sachs, *Über eine von H.S. Wilf angegebene Schranke für die chromatische Zahl endlicher Graphen*, Math. Nachr., 39 (1969), pp. 373–386.
[43] R. Fritsch and G. Fritsch, *The Four Color Theorem: History, Topological Foundations, and Idea of Proof*, Springer-Verlag, New York, 1998.
[44] A. Gamst, *Some lower bounds for a class of frequency assignment problems*, IEEE Trans. Vehicular Tech., 35 (1986), pp. 8–14.
[45] A. H. Gebremedhin, *Practical Parallel Algorithms for Graph Coloring Problems in Numerical Optimization*, Ph.D. thesis, University of Bergen, Norway, 2003.
[46] A. H. Gebremedhin and F. Manne, *Scalable parallel graph coloring algorithms*, Concurrency: Practice and Experience, 12 (2000), pp. 1131–1146.
[47] A. H. Gebremedhin, F. Manne, and A. Pothen, *Parallel distance-k coloring algorithms for numerical optimization*, in Euro-Par 2002 Parallel Processing, B. Monien and R. Feldmann, eds., Lecture Notes in Comput. Sci. 2400, Springer-Verlag, Berlin, 2002, pp. 912–921.
[48] A. H. Gebremedhin, A. Tarafdar, F. Manne, and A. Pothen, *New Acyclic and Star Coloring Algorithms with Application to Computing Hessians*, Tech. Report, Old Dominion University, Norfolk, VA, 2005.
[49] U. Geitner, J. Utke, and A. Griewank, *Automatic computation of sparse Jacobians by applying the method of Newsam and Ramsdell*, in Computational Differentiation: Techniques, Applications, and Tools, M. Berz, C. Bischof, G. Corliss, and A. Griewank, eds., SIAM, Philadelphia, 1996, pp. 161–172.
[50] R. Giering and T. Taminski, *Recipes for Adjoint Code Construction*, Tech. Report 212, Max-Planck Institut für Meteorologie, Hamburg, 1996.
[51] R. K. Gjertsen Jr., M. T. Jones, and P. E. Plassmann, *Parallel heuristics for improved, balanced graph colorings*, J. Parallel Distrib. Comput., 37 (1996), pp. 171–186.
[52] D. Goldfarb and P. L. Toint, *Optimal estimation of Jacobian and Hessian matrices that arise in finite difference calculations*, Math. Comp., 43 (1984), pp. 69–88.
[53] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, SIAM, Philadelphia, 2000.
[54] A. Griewank and G. F. Corliss, eds., *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, SIAM, Philadelphia, 1991.
[55] A. Griewank, D. Juedes, and J. Utke, *ADOL-C: A package for the automatic differentiation of algorithms written in C/C++*, ACM Trans. Math. Software, 22 (1996), pp. 131–167.
[56] B. Grünbaum, *Acyclic colorings of planar graphs*, Israel J. Math., 14 (1973), pp. 390–408.
[57] G. Hajós, *Über eine Konstruktion nicht n-färbbarer Graphen*, Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Nat. Reihe, 10 (1961), pp. 116–117.
[58] J. Hall, J. Hartline, A. R. Karlin, J. Saia, and J. Wilkes, *On algorithms for efficient data migration*, in Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2001, pp. 620–629.
[59] M. M. Halldórsson, *A still better performance guarantee for approximate graph coloring*, Inform. Process. Lett., 45 (1993), pp. 19–23.

[60] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed., Morgan-Kaufmann, San Francisco, 2002.
[61] S. Hossain, *On the Computation of Sparse Jacobian Matrices and Newton Steps*, Ph.D. thesis, Tech. Report 146, Department of Informatics, University of Bergen, Norway, 1998.
[62] S. Hossain and T. Steihaug, *Computing a sparse Jacobian matrix by rows and columns*, Optim. Methods Softw., 10 (1998), pp. 33–48.
[63] S. Hossain and T. Steihaug, *Reducing the number of AD passes for computing a sparse Jacobian matrix*, in Automatic Differentiation of Algorithms: From Simulation to Optimization, G. Corliss, C. Faure, A. Griewank, L. Hascoët, and U. Naumann, eds., Springer-Verlag, New York, 2002, pp. 263–270.
[64] S. Hossain and T. Steihaug, *Sparsity issues in computation of Jacobian matrices*, in Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, T. Mora, ed., ACM, New York, 2002, pp. 123–130.
[65] S. Hossain and T. Steihaug, *Optimal Direct Determination of Sparse Jacobian Matrices*, Tech. Report 254, Department of Informatics, University of Bergen, Norway, 2003.
[66] P. D. Hovland, *Personal communication*, 2004.
[67] P. D. Hovland, *Combinatorial problems in automatic differentiation*, presented at the SIAM Workshop on Combinatorial Scientific Computing, 2004.
[68] P. D. Hovland, *Automatic Differentiation of Parallel Programs*, Ph.D. thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 1997.
[69] P. D. Hovland and L. C. McInnes, *Parallel simulation of compressible flow using automatic differentiation and PETSc*, Parallel Comput., 27 (2001), pp. 503–519.
[70] D. Hysom and A. Pothen, *A scalable parallel algorithm for incomplete factor preconditioning*, SIAM J. Sci. Comput., 22 (2001), pp. 2194–2215.
[71] T. Jensen and B. Toft, *Graph Coloring Problems*, Wiley-Interscience, New York, 1995.
[72] D. S. Johnson, *Worst case behaviour of graph coloring algorithms*, Congr. Numer., 10 (1974), pp. 513–527.
[73] M. T. Jones and P. E. Plassmann, *A parallel graph coloring heuristic*, SIAM J. Sci. Comput., 14 (1993), pp. 654–669.
[74] M. T. Jones and P. E. Plassmann, *Scalable iterative solution of sparse linear systems*, Parallel Comput., 20 (1994), pp. 753–773.
[75] D. Karger, R. Motwani, and M. Sudan, *Approximate graph coloring by semidefinite programming*, J. ACM, 45 (1998), pp. 246–265.
[76] D. K. Kaushik, *Personal communication*, 2004.
[77] D. K. Kaushik, D. E. Keyes, and B. F. Smith, *On the interaction of architecture and algorithm in the domain-based parallelization of an unstructured grid incompressible flow code*, in Proceedings of the 10th International Conference on Domain Decomposition Methods, AMS, Providence, RI, 1998, pp. 311–319.
[78] D. E. Keyes, P. D. Hovland, L. C. McInnes, and W. Samyono, *Using automatic differentiation for second-order matrix-free methods in PDE-constrained optimization*, in Automatic Differentiation of Algorithms: From Simulation to Optimization, G. Corliss, C. Faure, A. Griewank, L. Hascoët, and U. Naumann, eds., Springer-Verlag, New York, 2001, pp. 35–50.
[79] D. A. Knoll and D. E. Keyes, *Jacobian-free Newton-Krylov methods: A survey of approaches and applications*, J. Comput. Phys., 193 (2004), pp. 357–397.
[80] A. V. Kosowski and K. Manuszewski, *Classical coloring of graphs*, in Graph Colorings, M. Kubale, ed., Contemp. Math. 352, AMS, Providence, RI, 2004, pp. 1–19.
[81] A. V. Kostochka, *Acyclic* 6*-coloring of planar graphs*, Metody Diskret. Analiz., 28 (1976), pp. 40–56 (in Russian).
[82] A. V. Kostochka, *Upper Bounds of Chromatic Functions of Graphs*, Ph.D. thesis, Novosibirsk, 1978 (in Russian).
[83] M. Krivelevich and B. Sudakov, *The chromatic number of random hypergraphs*, Random Structures Algorithms, 12 (1998), pp. 381–483.
[84] S. O. Krumke, M. V. Marathe, and S. S. Ravi, *Models and approximation algorithms for channel assignment in radio networks*, Wireless Networks, 7 (2001), pp. 567–574.
[85] M. Kubale, *Graph colouring*, in Encyclopedia of Microcomputers, Vol. 8, A. Kent and J. Williams, eds., Marcel Dekker, New York, 1991, pp. 47–69.
[86] M. Kubale, *Graph Colorings*, AMS, Providence, RI, 2004.
[87] S. L. Lee and P. D. Hovland, *Sensitivity analysis using parallel ODE solvers and automatic differentiation in C: SensPVODE and ADIC*, in Automatic Differentiation of Algorithms: From Simulation to Optimization, G. Corliss, C. Faure, A. Griewank, L. Hascoët, and U. Naumann, eds., Springer-Verlag, New York, 2001, Chap. 26, pp. 223–229.

[88] D. R. Lick and A. T. White, *k-degenerate graphs*, Canad. J. Math., 22 (1970), pp. 1082–1096.
[89] Y. L. Lin and S. S. Skiena, *Algorithms for square roots of graphs*, SIAM J. Discrete Math., 8 (1995), pp. 99–118.
[90] L. Lovász and M. D. Plummer, *Matching Theory*, North-Holland, Amsterdam, 1986.
[91] M. Luby, *A simple parallel algorithm for the maximal independent set problem*, SIAM J. Comput., 15 (1986), pp. 1036–1053.
[92] B. Manvel, *Extremely greedy coloring algorithms*, in Graphs and Applications, Proceedings of the 1st Colorado Symposium on Graph Theory, F. Harary and J. Maybee, eds., John Wiley, New York, 1985, pp. 257–270.
[93] D. W. Matula, *A min-max theorem for graphs with application to graph coloring*, SIAM Rev., 10 (1968), pp. 481–482.
[94] D. W. Matula, G. Marble, and J. Isaacson, *Graph coloring algorithms*, in Graph Theory and Computing, R. Read, ed., Academic Press, New York, 1972, pp. 109–122.
[95] D. W. Matula, Y. Shiloach, and R. E. Tarjan, *Two Linear-Time Algorithms for Five-Coloring a Planar Graph*, Tech. Report STAN-CS-80-830, Computer Science Department, Stanford University, Stanford, CA, 1980.
[96] D. W. Matula, Y. Shiloach, and R. E. Tarjan, *Analysis of two linear-time algorithms for five-coloring a planar graph*, Congr. Numer., 33 (1981), p. 407 (abstract).
[97] S. T. McCormick, *Optimal approximation of sparse Hessians and its equivalence to a graph coloring problem*, Math. Program., 26 (1983), pp. 153–171.
[98] J. Mitchem, *Every planar graph has an acyclic* 8*-coloring*, Duke Math. J., 41 (1974), pp. 177–181.
[99] R. Motwani, *Average-case analysis of algorithms for matchings and related problems*, J. ACM, 41 (1994), pp. 1329–1356.
[100] U. Naumann, *Cheaper Jacobians by simulated annealing*, SIAM J. Optim., 13 (2002), pp. 660–674.
[101] U. Naumann, *Optimal accumulation of Jacobian matrices by elimination methods on the dual computational graph*, Math. Program., 99 (2004), pp. 399–421.
[102] J. Neśetŕil and P. Ossan de Mendez, *Colorings and homomorphisms of minor closed classes*, in Discrete and Computational Geometry: The Goodman-Pollack Festschrift, B. Aronov, S. Basu, J. Pach, and M. Sharir, eds., Springer-Verlag, Berlin, 2003, pp. 651–664.
[103] G. N. Newsam and J. D. Ramsdell, *Estimation of sparse Jacobian matrices*, SIAM J. Algebraic Discrete Methods, 4 (1983), pp. 404–418.
[104] J. J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.
[105] W. Nooy, A. Mrvar, and V. Batageli, *Exploring Social Network Analysis with Pajek*, Cambridge University Press, Cambridge, UK, 2004.
[106] B. Norris, S. Balay, S. Benson, L. Freitag, P. D. Hovland, L. C. McInnes, and B. Smith, *Parallel components for PDEs and optimization: Some issues and experiences*, Parallel Comput., 28 (2002), pp. 1811–1831.
[107] M. J. D. Powell and P. L. Toint, *On the estimation of sparse Hessian matrices*, SIAM J. Numer. Anal., 16 (1979), pp. 1060–1074.
[108] P. Raghavan and E. Upfal, *Efficient routing in all-optical networks*, in Proceedings of the 26th Symposium on the Theory of Computing (STOC'94), ACM, New York, 1994, pp. 134–143.
[109] E. Ramadan, A. Tarafdar, and A. Pothen, *A hypergraph model for the yeast protein complex network*, in Proceedings of the IPDPS Workshop on High Performance Computational Biology, CD-ROM, IEEE, 2004.
[110] N. Rostaing, S. Dalmas, and A. Galligo, *Automatic differentiation in Odyssee*, Tellus, 45a (1993), pp. 558–568.
[111] Y. Saad, *ILUM: A multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput., 17 (1996), pp. 830–847.
[112] J. Schmidt-Pruzan, E. Shamir, and E. Upfal, *Random hypergraph coloring algorithms and the weak chromatic number*, J. Graph Theory, 8 (1985), pp. 347–362.
[113] G. Szekeres and H. S. Wilf, *An inequality for the chromatic number of a graph*, J. Combin. Theory, 4 (1968), pp. 1–3.
[114] C. Thomassen, *Every planar graph is* 5*-choosable*, J. Combin. Theory Ser. B, 62 (1994), pp. 180–181.
[115] B. Toft, *Coloring, stable sets and perfect graphs*, in Handbook of Combinatorics, Vol. I, R. L. Graham, M. Grötschel, and L. Lovász, eds., MIT Press, Cambridge, UK, 1996, pp. 233–288.

[116] M. Trick, *Network resources for coloring a graph*, http://mat.gsia.cmu.edu/COLOR/color.html.
[117] V. V. Vazirani, *Approximation Algorithms*, Springer-Verlag, Berlin, 2001.
[118] A. Walther, *ADOL-C: A Package for Automatic Differentiation of Algorithms Written in C/C++*, http://www.math.tu-dresden.de/wir/project/adolc/.
[119] G. Wegner, *Graphs with Given Diameter and a Coloring Problem*, Tech. Report, University of Dortmund, Germany, 1977.
[120] D. J. A. Welsh and M. B. Powell, *An upper bound for the chromatic number of a graph and its application to timetabling problems*, Comput. J., 10 (1967), pp. 85–86.
[121] A. Wigderson, *Improving the performance guarantee for approximate graph coloring*, J. ACM, 30 (1983), pp. 729–735.
[122] R. A. Wilson, *Graphs, Colourings, and the Four-Colour Theorem*, Oxford University Press, Oxford, 2002.
[123] L. A. Wolsey, *Integer Programming*, Wiley-Interscience, New York, USA, 1998.
[124] M. Yannakakis, *Computing the minimum fill-in is NP-complete*, SIAM J. Algebraic Discrete Methods, 2 (1981), pp. 77–79.