Graph Coloring in the Estimation of Sparse Derivative Matrices: Instances and Applications ⋆

Shahadat Hossain

Department of Mathematics and Computer Science, University of Lethbridge, Alberta, Canada

Trond Steihaug

Department of Informatics, University of Bergen, Bergen, Norway

Abstract

We describe a graph coloring problem associated with the determination of mathematical derivatives. The coloring instances are obtained as intersection graphs of row partitioned sparse derivative matrices. The size of the graph is dependent on the partition and can be varied between the number of columns and the number of nonzero entries. If solved exactly our proposal will yield a significant reduction in computational cost. Coloring results from backtrack DSATUR and Small-k algorithms applied on the coloring instances are given. We analyze the test results and remark on the hardness of the generated coloring instances.

Key words: Exact Coloring, Sparsity, Mathematical Derivatives

1 Introduction

Computation or estimation of the Jacobian matrix $J \equiv \left\{ \frac{\partial f_i}{\partial x_j} \right\}$ of a mapping $f : \mathbf{R}^n \to \mathbf{R}^m$ constitutes an important subtask in many numerical procedures. Mathematical derivatives can be approximated or calculated by a variety of techniques including automatic (or algorithmic) differentiation (AD) [12], finite differencing (FD) [7], and computational divided differencing (CDD) (or algorithmic differencing) [22].

⋆ Presented at ISMP2003

Email addresses: `hossain@cs.uleth.ca` (Shahadat Hossain), `trond.steihaug@ii.uib.no` (Trond Steihaug).

URLs: `http://www.cs.uleth.ca/~hossain` (Shahadat Hossain), `http://www.ii.uib.no/~trond` (Trond Steihaug).

The derivative matrices often posses exploitable information such as sparsity and other special structurer like symmetry. Numerical techniques for solving such large-scale problems must obtain these derivative information efficiently and accurately. For a sparse matrix with known sparsity pattern or if the sparsity can be determined easily [13] substantial savings in the computational cost can be achieved.

A group of columns in which no two columns have nonzero elements in the same row position is known as *structurally orthogonal*. If columns j and k are structurally orthogonal, then for each row index i at most one of $J(i,j)$ and $J(i,k)$ can be nonzero. In general $\sum_j J(i,j) = J(i,k)$ for some k (k will depend on i) where the sum is taken over a group of structurally orthogonal columns. An estimate of the nonzero elements in the group can be obtained in

$$\left.\frac{\partial f(x+ts)}{\partial t}\right|_{t=0} = f'(x)s \approx As = \frac{1}{\varepsilon}[f(x+\varepsilon s) - f(x)] \equiv b \tag{1}$$

with a forward difference (one extra function evaluation) where b is the finite difference approximation and $s = \sum_j e_j$. With forward automatic differentiation the unknown elements in the group are obtained as the product $b = f'(x)s$ accurate up to the round-off error resulting from the finite machine precision. The matrix is completely determined from a structurally orthogonal partition of the columns into $p \leq n$ groups.

The following notational conventions are used in the paper. If an uppercase letter is used to denote a matrix (e.g., A), then the (i,j) entry is denoted by $A(i,j)$ or by corresponding lowercase letter a_{ij}. We also use colon notation [11] to specify a submatrix of a matrix. For $A \in R^{m\times n}$, $A(i,:)$ and $A(:,j)$ denotes the ith row and the jth column respectively. For a vector of column indices v, $A(:,v)$ denotes the submatrix consisting of columns whose indices are contained in v. For a vector of row indices u, $A(u,:)$ denotes the submatrix consisting of rows whose indices are contained in u. A vector is specified using only one dimension. For example, the ith element of $v \in R^n$ is written $v(i)$. The transpose operator is denoted by $(\cdot)^T$. A blank or "0" represents a zero entry and any other symbol in a matrix denotes a nonzero entry.

The matrix determination problem can be conveniently modelled by graphs [3,17,21] and such graph models often reveal valuable properties that can be utilized in finding efficient solutions. It has been observed that finding a partition of the columns of A in groups of structurally orthogonal columns is equivalent to coloring the vertices of the associated column intersection graph. A combined approach where sparsity is exploited more effectively by a bi-directional partitioning scheme has been proposed in [5,15]. These techniques use the forward and reverse modes of AD to compute the products AV and W^TA for matrices V and W. Our recent proposal [17] shows that sparsity information can be exploited more effectively by partitioning column segments. A graph coloring formulation of this partitioning problem has also been considered there. In this paper we present a practical application of the

column segment approach and describe the coloring instances and analyze their properties. The specific contributions of this work are as follows.

(1) The column segments method generalizes the column partitioning problem in sparse derivative matrix estimation. Its equivalence to a graph coloring problem provides benchmark coloring instances some of which are hard to solve. Furthermore, standard benchmark matrix instances e.g., Harwell-Boeing [9] collection constitute a rich set of real-life data that can be used to generate practical coloring instances. We have developed C++ code *CsegGraph* [18] that implements the graph generator [1] .
(2) Our logarithmic example represents general structure of practical problems from the field of molecular distance geometry. We show that the computational effort (measured in terms of the number of AD forward mode evaluations or the number of function evaluations) in the calculation of first derivative vector of the underlying function is logarithmic in the number of independent variables n. This is a considerable gain in computational efficiency compared with ordinary column partition which needs n forward mode evaluations or function evaluations.
(3) Our test results from the Harwell-Boeing benchmark problems are obtained by two well-known exact coloring algorithm e.g., back-track DSATUR, Small-k, and a SAT solver Chaff [10]. Chaff outperformed the other two and was able to solve all 4 benchmark instances. We provide detail analysis on the performance of the three solvers.

The remainder of the paper is organized as follows. In section 2 we review a partitioning scheme [17] based on structurally orthogonal column segments to determine sparse Jacobian matrices. A graph coloring formulation of the partitioning problem is given. We show that the coloring problem is unlikely to be solved efficiently. Complexity results concerning the colorability of the column-segment graph is given. Section 3 presents the molecular distance geometry problem where the column segment method yields the gradient of the objective function efficiently. Section 4 contains results from computational testing of coloring instances obtained from our column segments graph generator. The test results from backtrack DSATUR [1] and Small-k [6] algorithms are presented and analyzed. Finally, the paper is concluded in section 5 with a discussion on topics for further studies.

2 Column Segment Partitioning and Graph Coloring

We assume that the sparsity pattern of the Jacobian matrices considered in this paper is known a priori. Also, the whole column is computed even if only part of a

[1] A copy can be obtained by contacting the first author at:
shahadat.hossain@uleth.ca

column is needed.

Both finite difference approximation and automatic differentiation allows the determination of a Jacobian matrix from its product with a specific set of vectors. Then the problem of sparse Jacobian matrix determination can be stated as:

Obtain vectors $s_1, \ldots, s_p$ such that the matrix-vector products

$$b_i \equiv As_i,\ i = 1, \ldots, p \text{ or } B \equiv AS$$

determine the $m \times n$ matrix A uniquely. Matrix S is called the *seed matrix*. Note that once the p products are computed and stored in B the unknown entries of matrix A in row $i, i = 1, 2, \ldots, m$ are obtained by solving the following m sets of linear equation systems:

$$AS = B.$$

The seed matrix S determines the structure of the linear systems to be solved for

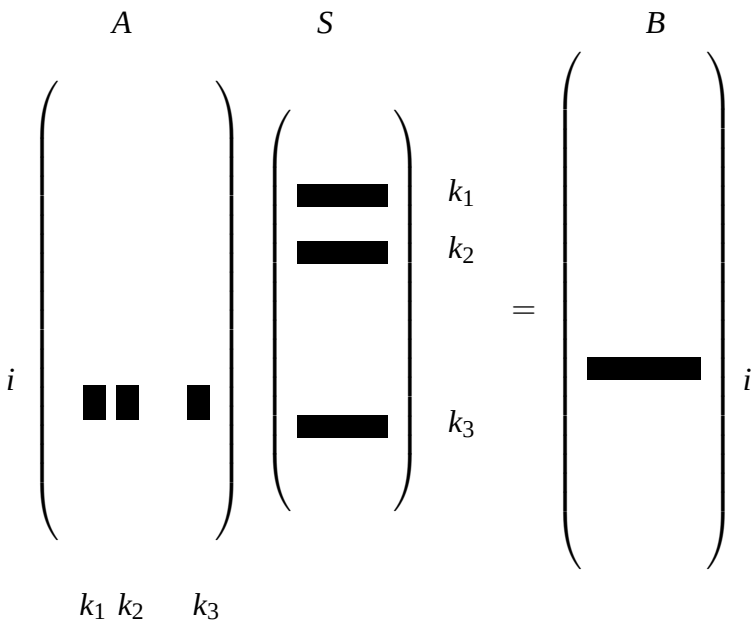

Fig. 1. Determination of the unknown entries with column indices k_1, k_2, and k_3 in row i of A.

the unknown elements in each row of A. In Figure 1 row i (number of nonzero elements in row i, $\rho_i = 3$) of the matrix A are to be determined.

The Jacobian matrix is said to be *determined directly* if S is such that the unknown entries in each row can be read-off (i.e., with no extra arithmetic operation except memory read) from the (reduced) linear system. A precise characterization of optimal direct determination of sparse Jacobian matrices can be found in [17]. Efficient direct determination is concerned with finding seed matrices with fewest columns.

As we shall see in this paper the efficient direct determination of Jacobian matrices is closely related with graph coloring problems and that the coloring instances for certain test problems can be very hard to solve.

A *Graph* $G = (V,E)$ is a set V of *vertices* and a set E of *edges*. An edge $e \in E$ is denoted by an unordered pair $\{u,v\}$ which connects vertices u and v, $u,v \in V$. A graph G is said to be a *complete* graph or *clique* if there is an edge between every pair of distinct vertices. In this paper multiple edges between a pair of vertices are considered as a single edge. A *p-coloring* of the vertices of G is a function $\Phi : V \to \{1,2,\cdots,p\}$ such that $\{u,v\} \in E$ implies $\Phi(u) \neq \Phi(v)$. *The chromatic number* $\chi(G)$ of G is the smallest p for which it has a *p-coloring*. An *optimal coloring* is a *p-coloring* with $p = \chi(G)$. Given an $m \times n$ matrix A, the *intersection graph* of the columns of A is denoted by $G(A) = (V,E)$ where corresponding to $A(:,j), j = 1,2,\ldots,n$, there is a vertex $v_j \in V$ and $\{v_j, v_l\} \in E$ if and only if $A(:,j)$ and $A(:,l), l \neq j$ have nonzero elements in the same row position.

Let Π be a partition of $\{1,2,\ldots,m\}$ yielding $w_1, w_2, \ldots, w_q$ where $w_{\tilde{i}}$ contains the row indices that constitute block $\tilde{i}$ and $A(w_{\tilde{i}},:) \in R^{m_{\tilde{i}} \times n}$, $\tilde{i} = 1,2,\ldots,q$. A segment of column j in block $\tilde{i}$ of A denoted by $A(w_{\tilde{i}}, j), \tilde{i} = 1,2,\ldots,q$ is called a *column segment*.

Definition 2.1 *Structurally orthogonal column segment*

- (`Same Column`)
  Column segments $A(w_{\tilde{i}}, j)$ and $A(w_{\tilde{k}}, j)$, $\tilde{i} \neq \tilde{k}$ are *structurally orthogonal*
- (`Same Row Block`)
  Column segments $A(w_{\tilde{i}}, j)$ and $A(w_{\tilde{i}}, l)$, $j \neq l$ are *structurally orthogonal* if they do not have nonzero entries in the same row position.
- (`Different`)
  Column segments $A(w_{\tilde{i}}, j)$ and $A(w_{\tilde{k}}, l)$, $\tilde{i} \neq \tilde{k}$ and $j \neq l$ are *structurally orthogonal* if
  · $A(w_{\tilde{i}}, j)$ and $A(w_{\tilde{i}}, l)$ are *structurally orthogonal* and
  · $A(w_{\tilde{k}}, j)$ and $A(w_{\tilde{k}}, l)$ are *structurally orthogonal*

An *orthogonal partition of column segments* is a mapping

$$\kappa : \{(\tilde{i}, j) : 1 \leq \tilde{i} \leq q, 1 \leq j \leq n\} \to \{1,\ldots,p\}.$$

where column segments in each group are structurally orthogonal. A sparse Jacobian matrix can be directly determined from an orthogonal partition of the column segments [17]. Let κ be any orthogonal partition of the column segments in p groups. Then the seed matrix defined by partition κ is

$$S(:,j) = \sum_{\{k:\kappa(\tilde{i},k)=j, 1 \leq \tilde{i} \leq q\}} e_k.$$

where e_k is the kth coordinate vector.

Definition 2.2 Given matrix A and row q-partition Π, the *column-segment graph* associated with A under partition Π is a graph $G_\Pi(A) = (V,E)$ where the vertex $v_{\tilde{i}j} \in V$ corresponds to the column segment $A(w_{\tilde{i}}, j)$ not identically 0, and $\{v_{\tilde{i}j}, v_{\tilde{k}l}\} \in E$ $1 \leq \tilde{i}, \tilde{k} \leq q, 1 \leq j, l \leq n$ if and only if column segments $A(w_{\tilde{i}}, j)$ and $A(w_{\tilde{k}}, l)$ are not structurally orthogonal.

The problem of determining Jacobian matrices using column segments can be stated as the following graph problem.

Theorem 1 *[17] Φ is a coloring of $G_\Pi(A)$ if and only if Φ induces a orthogonal partition κ of the column segments of A.*

This special graph coloring problem, however, is no easier than the general graph p-coloring. Let $|\cdot|$ denote the number of elements contained in a set.

Lemma 1 *[17] Given a graph $G = (V,E)$, and positive integers $1 \leq p \leq |V| = n$ and $1 \leq q$ there is a graph $\overline{G} = (\overline{V}, \overline{E})$ such that $\overline{G}$ is p-colorable if and only if G is p-colorable.*

To show that Jacobian estimation by a direct method is no easier than the general graph coloring problem it suffices to show that $\overline{G}$ is isomorphic to column segment graph $G_\Pi(A)$ for some Jacobian matrix A. Let $H = (V,E)$ be a graph where $V = \{v_1, v_2, \cdots, v_n\}$ and $E = \{e_1, e_2, \cdots, e_m\}$. Define $\mathcal{H} \in R^{m \times n}$ such that $e_i = \{v_k, v_l\}$ implies

$$\mathcal{H}(i,k) \neq 0, \mathcal{H}(i,l) \neq 0 \text{ and for } j \neq k \neq l \ \mathcal{H}(i,j) = 0.$$

Then it can be shown that H is isomorphic to $G(\mathcal{H})$.

Theorem 2 *[17] Given a graph $H = (V,E)$ and positive integers $p \leq n$ and $q \leq m$ there exists a mapping $f : R^{np} \to R^{m+n}$ such that for a row q-partition Π of $f'(x)$, $G_\Pi(f'(x))$ is isomorphic to $\overline{H}$.*

In Theorem 2, $\overline{H}$ is obtained from H as defined by Lemma 1. Theorems 1 and 2 give the following result.

Theorem 3 *Given a matrix A finding a minimum coloring for $G_\Pi(A)$ is NP-hard.*

Let Π_m and Π_1 denote the row m-partition and row 1-partition respectively. The number of blocks k defined by the row partition Π varies from 1 to m, with Π_1 resulting in the column intersection graph (Theorem 2.1 [3]) while Π_m yielding the *element isolation graph* of A [21].

A *refinement* of row q-partition Π yields a row q'-partition Π' where the row blocks of Π are subdivided further so that $q \leq q'$.

Theorem 4 *For a row q'-partition Π' resulting from any refinement of row q-*

partition Π

$$\chi(G_{\Pi'}(A)) \leq \chi(G_{\Pi}(A)).$$

For all directly determined matrices the number of columns p of the seed matrix S satisfies

$$p \geq \chi(G_{\Pi_m}(A)) \quad [17].$$

The following inequalities follow from Theorem 4.

Corollary 1 $\chi(G_{\Pi_m}(A)) \leq \chi(G_{\Pi'}(A)) \leq \chi(G_{\Pi}(A)) \leq \chi(G_{\Pi_1}(A)) = \chi(G(A))$

It is clear that row partitioning determines the size of $G_{\Pi}(A)$. Although $G_{\Pi}(A)$ can be larger than $G(A)$, $G_{\Pi}(A)$ never requires more colors than $G(A)$. Therefore, an important question is if there is an automatic way to construct a row partition Π_k with $k < m$ blocks such that $\chi(G_{\Pi_k}(A))$ is equal (or very close) to $\chi(G_{\Pi_m}(A))$. The rows are (structurally) orthogonal if the columns of A^T are (structurally) orthogonal. The following result shows that any partitioning $\widehat{\Pi}$ in groups of structural orthogonal columns of A^T can be used.

Theorem 5 *For any orthogonal column partition* $\hat{\Pi}$ *of* A^T, $\chi(G_{\Pi_m}(A)) = \chi(G_{\hat{\Pi}}(A))$.

As a consequence of Theorem 5 we observe that a consistent partition (one that is found by some heuristic) rather than an optimal partition (i.e. optimal coloring of $G(A^T)$) of matrix A^T would suffice.

3 The Log-example and the Molecular Conformation Problem

The molecular conformation problem in cluster statics is concerned with the determination of the minimum energy configuration of a cluster of atoms or molecules [20]. This problem is formulated as a unconstrained minimization problem where the objective function can be represented in a partially separable (see [14]) form. The gradient of partially separable functions can be computed efficiently.

Our log-example represents a class of sparsity patterns that are specially hard for determination methods based on column partitioning. The difficulty in exploiting sparsity in those examples is reflected in coloring the associated column intersection graph which are complete graphs. With column segments the available sparsity is exploitable as the associated graph is no longer a complete graph.

3.1 The Log-example

The central idea in the definition of log-example instances is best described in terms of recursive definition of complete bipartite graphs. Let G^0_{k-1} and G^1_{k-1} be complete bipartite graphs on $2^{(k-1)}$, $k \geq 1$ vertices each. Then $G_k \equiv G^0_k$ is the complete bipartite graph on 2^k vertices where $V_k = V^0_{k-1} \cup V^1_{k-1}$ and $E_k = \{\{u,v\} : u \in V^0_{k-1}, v \in V^1_{k-1}\}$. Note that $G^{\hat{i}}_0 = (V^{\hat{i}}_0, E^{\hat{i}}_0)$ with $V^{\hat{i}}_0 = \{v_i\}$, $E^{\hat{i}}_0 = \emptyset$, for $i, \hat{i} = 0, 1, \ldots, (2^k - 1)$.

Given graph $G = (V,E)$ where $V = \{v_0, v_1, \ldots, v_{n-1}\}$ and $E = \{e_0, e_1, \ldots, e_{m-1}\}$ we can define $m \times n$ matrix A such that $e_l = \{v_i, v_j\} \in E$ if and only if $A(l,i)$, $A(l,j) \neq 0$ for indices i, j, l. It then follows that $G(A)$ is isomorphic to G.

Let $k \geq 2$ and let $G = (V,E)$, $V = \bigcup_{\hat{i}=0}^{(2^k-1)} V^{\hat{i}}_0$, $E = \bigcup_{\hat{i}=0}^{(2^k-1)} E^{\hat{i}}_0 = \emptyset$
For $\tilde{i} = 0, 1, \ldots, (k-1)$ **do**
  Let $i_{k-1} i_{k-2} \ldots i_0$ and $j_{k-1} j_{k-2} \ldots j_0$ be the binary representations of i and j for $i, j = 0, 1, \ldots, (2^k - 1)$, respectively. Define $\{v_i, v_j\} \in E^{\hat{i}}_{\tilde{i}}$ for some $\hat{i}$ if $i_{\tilde{i}} \neq j_{\tilde{i}}$ and $i_{\tilde{l}} = j_{\tilde{l}}$ for all $\tilde{l} > \tilde{i}$.
  Set $E = E \bigcup (\bigcup_{\hat{i}=0}^{2^{k-\tilde{i}-1}-1} E^{\hat{i}}_{\tilde{i}})$.
End-For

As noted earlier an edge $\{v_i, v_j\}$ of G can be represented by a row containing two nonzero entries in columns i and j of $2^k \times 2^{k-1}(2^k - 1)$ matrix A such that $G(A) = G$.

A row partition (or coloring) can be defined by according to the algorithm where the $\tilde{i}$'th block $A_{\tilde{i}}$ consists of rows defined in step $\tilde{i}$. Then there are k blocks in the row partition. A suitable seed matrix for direct determination of nonzero entries of a Jacobian matrix with log-example sparsity pattern is defined by

$$S(:, 2\tilde{i}) = \sum_{\{i : i_{\tilde{i}} = 0, i = (i_{k-1} i_{k-2} \ldots i_{\tilde{i}} \ldots i_0)_2\}} e_i \text{ and } S(:, 2\tilde{i} + 1) = \sum_{\{i : i_{\tilde{i}} \neq 0, i = (i_{k-1} i_{k-2} \ldots i_{\tilde{i}} \ldots j_0)_2\}} e_i \tag{2}$$

for $\tilde{i} = 0, 1, \ldots, (k-1)/2$ and $(\cdot)_2$ denotes the binary representation of the number in the parentheses. This implies that the number of columns in S is $2k$. With this seed matrix A can be determined directly using only $2k$ forward mode AD passes. Since $G(A)$ is a complete graph, 2^k forward passes are necessary to compute A without row partitioning. The above procedure can be generalized to define matrices where basic unit of construction in $G(A)$ is a clique on a constant number of vertices. Figure 2 illustrates the sparsity pattern for a log-example on 8 columns. The seed matrix S is obtained from a row 3-partition ($k = 3$) defined by Equation (2).

$$A = \begin{pmatrix}
\times & \times & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \times & \times & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \times & \times & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \times & \times \\
\times & 0 & \times & 0 & 0 & 0 & 0 & 0 \\
\times & 0 & 0 & \times & 0 & 0 & 0 & 0 \\
0 & \times & \times & 0 & 0 & 0 & 0 & 0 \\
0 & \times & 0 & \times & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \times & 0 & \times & 0 \\
0 & 0 & 0 & 0 & \times & 0 & 0 & \times \\
0 & 0 & 0 & 0 & 0 & \times & \times & 0 \\
0 & 0 & 0 & 0 & 0 & \times & 0 & \times \\
\times & 0 & 0 & 0 & \times & 0 & 0 & 0 \\
\times & 0 & 0 & 0 & 0 & \times & 0 & 0 \\
\times & 0 & 0 & 0 & 0 & 0 & \times & 0 \\
\times & 0 & 0 & 0 & 0 & 0 & 0 & \times \\
0 & \times & 0 & 0 & \times & 0 & 0 & 0 \\
0 & \times & 0 & 0 & 0 & \times & 0 & 0 \\
0 & \times & 0 & 0 & 0 & 0 & \times & 0 \\
0 & \times & 0 & 0 & 0 & 0 & 0 & \times \\
0 & 0 & \times & 0 & \times & 0 & 0 & 0 \\
0 & 0 & \times & 0 & 0 & \times & 0 & 0 \\
0 & 0 & \times & 0 & 0 & 0 & \times & 0 \\
0 & 0 & \times & 0 & 0 & 0 & 0 & \times \\
0 & 0 & 0 & \times & \times & 0 & 0 & 0 \\
0 & 0 & 0 & \times & 0 & \times & 0 & 0 \\
0 & 0 & 0 & \times & 0 & 0 & \times & 0 \\
0 & 0 & 0 & \times & 0 & 0 & 0 & \times
\end{pmatrix}, \quad S = \begin{pmatrix}
\times & 0 & \times & 0 & \times & 0 \\
0 & \times & \times & 0 & \times & 0 \\
\times & 0 & 0 & \times & \times & 0 \\
0 & \times & 0 & \times & \times & 0 \\
\times & 0 & \times & 0 & 0 & \times \\
0 & \times & \times & 0 & 0 & \times \\
\times & 0 & 0 & \times & 0 & \times \\
0 & \times & 0 & \times & 0 & \times
\end{pmatrix}$$

Fig. 2. Sparsity structure of log-example on 8 columns and an associated seed matrix.

3.2 Molecular Conformation Problem

The molecular conformation problem in cluster statics is concerned with the determination of the minimum energy configuration of a cluster of atoms or molecules [20]. This problem is formulated as a unconstrained minimization problem where the objective function can be represented in a partially separable (see [14]) form. The gradient of a partially separable function can be computed efficiently.

Given the positions $p_1, p_2, \ldots, p_n$ of n molecules in R^d, the energy potential is defined as

$$\sum_{j=1}^{n} \sum_{i=1}^{j-1} \phi(\|p_j - p_i\|_2), \tag{3}$$

where $\phi : R \to R$ is the potential function between pairs of molecules. We assume that $\phi(r)$ is differentiable for $r \geq 0$. The molecular conformation problem is to find

$p = (p_1, p_2, \ldots, p_n) \in R^{nd}$ such that the potential (3) is minimized.

Consider the molecular conformation problem in plane i.e., $d = 2$. Let the coordinates for position p_i be x_{2i-1} and x_{2i}. Then we have $2n$ independent variables $x_1, \ldots, x_{2n}$. It follows that ϕ can be written as a function of 4 variables $\psi_{ij}(x_{2i-1}, x_{2i}, x_{2j-1}, x_{2j})$ for positions p_i and p_j. Let

$$\Psi_2(x) = \sum_{j=1}^{n} \sum_{i=1}^{j-1} \psi_{ij}(x), \tag{4}$$

Then the gradient of Ψ_2 is

$$\nabla \Psi_2(x) = \tilde{\Psi}'(x)^T e, \tilde{\Psi}(x) = (\psi_{21}(x)\, \psi_{31}(x)\, \psi_{32}(x) \ldots \psi_{nd\,nd-1}(x))^T \tag{5}$$

where $\tilde{\Psi}'(x)$ is the Jacobian matrix of $\tilde{\Psi}$ at x. It is easily seen that each row of the Jacobian matrix $\tilde{\Psi}'(x)$ contains 4 nonzero elements corresponding to the variables $x_{2i-1}, x_{2i}, x_{2j-1},$ and x_{2j}.

Although $\tilde{\Psi}'(x)$ is sparse (only 4 nonzero elements per row), it may not be possible to exploit the sparsity directly since corresponding intersection graph is a complete graph. We show that the Jacobian matrix $\tilde{\Psi}'(x)$ can be computed efficiently by identifying its sparsity structure with our log-example of appropriate dimension.

Let $G = (V, E)$ be a graph and let $e \in E$ with $e = \{u, v\}$. The *contraction* of e in G is the operation that removes e and the vertices u and v, and adds a new vertex w to the graph such that w is connected with exactly those vertices that were connected with u or v in G. An edge contraction in G is the contraction of some edge $e \in G$. Note that a sequence of edge contractions may lead to multiple edges between a pair of vertices in the resulting graph. In such a case we ignore the multiple edges and treat them as a single edge.

Consider the Jacobian matrix $\tilde{\Psi}'(x)$. For simplicity we assume that $n = 2^k d$, $k \geq 2$. Corresponding to the component function ψ_{ij} there is a row in $\tilde{\Psi}'(x)$ containing nonzero elements in columns $x_{2i-1}, x_{2i}, x_{2j-1}, x_{2j}$. In the intersection graph $G(\tilde{\Psi}'(x))$ vertices $x_{2i-1}, x_{2i}, x_{2j-1}, x_{2j}$ constitutes a clique. For each such clique we contract edges $\{x_{2i-1}, x_{2i}\}$ and $\{x_{2j-1}, x_{2j}\}$. This corresponds to the columns $\{x_{2i-1}, x_{2i}\}$ and $\{x_{2j-1}, x_{2j}\}$ being "collapsed" into columns p_i and p_j respectively. Note that the columns x_{2i-1} and x_{2i} for $i = 1, 2, \ldots, n$ have the same sparsity structure and the collapsing is done in a structural sense. If we reorder the rows, it is clear that the above construction gives the log-example.

To actually compute $\tilde{\Psi}'(x)$ we need 4 AD forward passes for each block instead of 2. This construction easily carries over to three or more dimensions.

If the number of molecules n satisfies

$$\frac{n}{\lceil log_2 n \rceil} > 2d \tag{6}$$

then the number of AD forward passes to compute the Jacobian matrix is strictly less than n.

4 The Column Segment Graph Generator

In this section we describe an algorithm for constructing *column segment graph* $G_\Pi(A)$ associated with a $m \times n$ matrix A given row partition Π. Furthermore, we describe the column segment matrix associated with the given row partition.

Let Π be a row partition of matrix A that partitions the rows into blocks $A_1, A_2, \cdots A_k$ (See Fig. 3(a)). Denote the intersection graph corresponding to $A_{\tilde{i}}$ by $G(A_{\tilde{i}}), \tilde{i} = 1, 2, \cdots k$. The construction of A_Π involves two phases. In the first phase, blocks $A_{\tilde{i}}$, $\tilde{i} = 1, 2, \cdots, k$ are placed successively in left to right fashion (see Fig. 3(b)) such that each nonzero column segment is mapped to a unique column of A_Π. In other words, for every nonzero column segment of A, a column is created in A_Π where all the entries are zero except that the column segment is copied in the matching row positions. This situation is illustrated in the top part of Fig. 3(b). In the second phase of

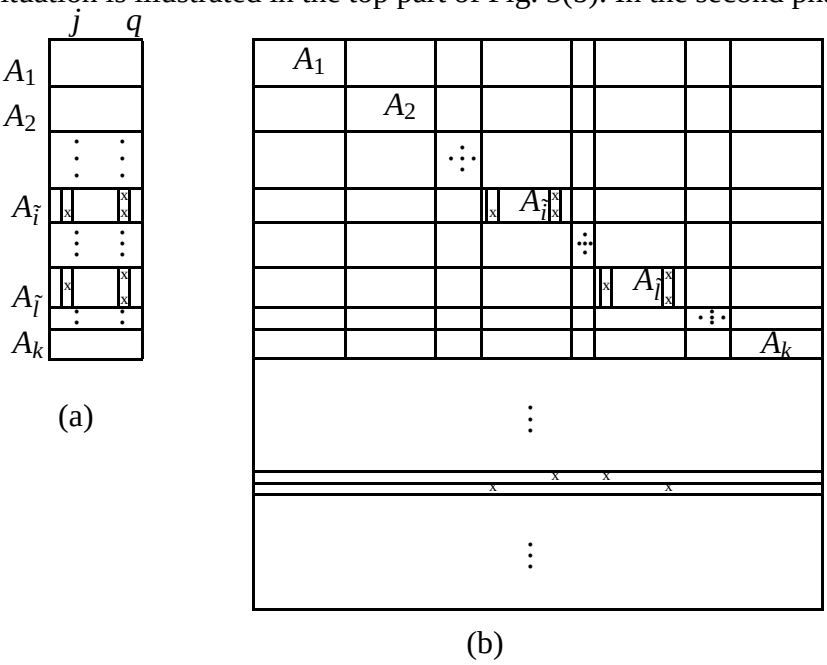

Fig. 3. (a): Matrix A is partitioned into k blocks (b): Column segment matrix corresponding to the partition.

construction, restrictions are enforced on column segments that are not structurally orthogonal in order to prevent them from being grouped together. Consider column

segments $A(w_{\tilde{i},j})$ and $A(w_{\tilde{i},q})$ in $A_{\tilde{i}}$. If there are nonzero entries in the same row position in $A(w_{\tilde{i},j})$ and $A(w_{\tilde{i},q})$ then they are not orthogonal implying that $A(w_{\tilde{i},j})$ is not orthogonal to column segments $A(w_{\tilde{p}})$ for all $p \neq \tilde{i}$. Consequently, $A(w_{\tilde{i},j})$ cannot be grouped together with any of the segments $A(w_{\tilde{p},q})$. Similarly, $A(w_{\tilde{i},q})$ is not orthogonal to columns $A(w_{\tilde{p},j})$ for all $\tilde{p} \neq \tilde{i}$. To enforce these restrictions we simply introduce two new rows in A_{Π}, one containing nonzero entries in the column positions mapped by the column segments $A(w_{\tilde{i},j})$ and $A(w_{\tilde{p},q})$ and the other containing nonzero entries in the column positions mapped by the column segments $A(w_{\tilde{i},q})$ and $A(w_{\tilde{p},j})$ for all $\tilde{p} \neq \tilde{i}$. This is done for every pair of dependent column segments in $A_{\tilde{i}}$, $\tilde{i} = 1, 2, \cdots, k$ (see Fig. 3(b)). To see this dependency restriction in terms of graphs, consider vertices $v_{\tilde{i}j}$ and $v_{\tilde{i}q}$ in $G(A_{\tilde{i}})$. For each such edge we define edges between vertex $v_{\tilde{i}j}$ and vertices $v_{\tilde{p}q}$ from $G(A_{\tilde{p}})$ for $\tilde{p} \neq \tilde{i}$. Similarly, vertex $v_{\tilde{i}q}$ is connected with the vertices $v_{\tilde{p}j}$ from $G(A_{\tilde{p}})$ for $\tilde{p} \neq \tilde{i}$. This situation is illustrated in Fig. 4. In Fig. 3(a) the matrix is partitioned into k blocks denoted

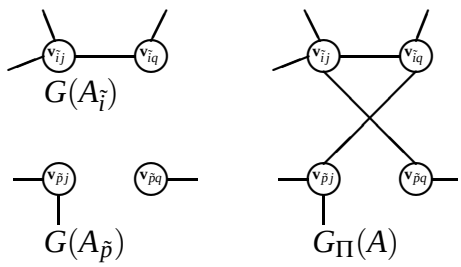

Fig. 4. Graph $G_{\Pi}(A)$ before and after the insertion of edges due to the edge $\{v_{\tilde{i}j}, v_{\tilde{i}q}\}$ in $G(A_i)$.

by $A_1, A_2, \cdots, A_k$. In Fig. 3(b) placement of each of the blocks $A_1, A_2, \cdots, A_k$ is shown. Column segments $A(w_{\tilde{i},j})$ and $A(w_{\tilde{i},q})$ are not orthogonal, and hence two rows containing nonzero entries in the appropriate columns are introduced.

An upper bound on the size of the column segment matrix and graph is easily obtained from its construction. For a k block partition, the number of columns $n' \leq \rho \leq n * k$ where ρ is the number of nonzero elements in A. The number of rows $m' \leq m + (k-1)\sum_{i=1}^{m} \rho_i(\rho_i - 1)$ where ρ_i is the number of nonzero in the ith row of A. In practice, however, the numbers n' and m' are smaller due to many zero column segments and repeated edges between pair of distinct vertices.

5 Graph Generator Implementation and Computational Experiments

Our graph generator is an object-oriented implementation of column segment graph instances using C++. This software uses SparseLib++v.1.5d [8], a collection of C++ sparse matrix classes that can read and convert between a number of standard

sparse matrix data structures e.g., coordinate, compressed column, and compressed row format which are also supported by Harwell-Boeing test matrix collection. The software also allows to test the graph instances using user supplied coloring algorithms. Currently, DSATUR and Small-k coloring routines are available with the package. Furthermore, two specialized classes of graph instances, the Eisenstat example and log-example, are provided. Detail information on the software can be found in [18].

The following example is due to Stanley Eisenstat[3]. Let A be a $(n+1)\times n$ matrix

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \text{ where } A_1 = [D_1 D_2] \text{ and } A_2 = \begin{bmatrix} Z & 0 \\ D_3 & B \end{bmatrix}$$

and $D_1, D_2, D_3 \in \mathbf{R}^{\frac{n}{2}\times\frac{n}{2}}$ are non-singular diagonal matrices. The diagonal entries of $B \in \mathbf{R}^{\frac{n}{2}\times\frac{n}{2}}$ are zeros while off-diagonal entries are nonzero, $Z \in \mathbf{R}^{1\times\frac{n}{2}}$ consists entirely of nonzero elements and $0 \in \mathbf{R}^{1\times\frac{n}{2}}$ is a zero vector. Further, $n \geq 6$ and n is an even number. From its construction a row 2-partition can be defined corresponding to first $n/2$ rows in row block 1 and the next $n/2+1$ rows in row block 2 to determine A with a total of $p = n/2+2$ AD forward mode evaluations. Table 1 presents the result of applying DSATUR coloring routine on Eisenstat examples. In the table n denotes the size of the matrix, k denotes the number of blocks in row partition, p denotes that the matrix rows are randomly permuted before defining row partition and u denotes no row permutation. As expected, the column segment matrix gets larger and denser with finer row partition. Row permutation seem to yield a smaller graph with a larger chromatic number for row partitions of the same cardinality. Note that different row permutation in general will yield different column segment graph. For larger problems DSATUR is unable to confirm the lower bound which is indicated by asterisk(*). The effect of permutation is also more prominent on the chromatic number as the size of the problem grows. On the other hand finer row partition does not yield a proportionate reduction in the chromatic number.

The experimental results from log-example displayed the similar pattern with regard to the effect of row partition and permutation on the chromatic number. Small-k coloring is best suited for graphs with small chromatic number. Since the chromatic number of both Eisenstat and log-example depends on the problem dimension, Small-k was unsuitable for larger instances. For both Eisenstat and log-example, only $k = m$ partition yielded the optimal coloring. The smallest log-example tested was the 28×8 matrix shown in Figure 5. The column segment graph with $k = m$ row partition has 56 vertices and 364 edges. Using the predefined row partition as in §3.1(3 blocks) it needed 6 colors while with $k = m$ row partition (28 blocks) the chromatic number of the resulting graph was found to be 5.

In Table 2 we report the following information. “Nodes” denotes the number of nodes, “Edges” denotes the number of edges, and $\chi(\cdot)$ denotes the chromatic number of $G(A)$ and $G_\Pi(A)$. Most striking difference in performance among the three

$$
\begin{pmatrix}
\times & \times & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \times & \times & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \times & \times & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \times & \times \\
\times & 0 & \times & 0 & 0 & 0 & 0 & 0 \\
\times & 0 & 0 & \times & 0 & 0 & 0 & 0 \\
0 & \times & \times & 0 & 0 & 0 & 0 & 0 \\
0 & \times & 0 & \times & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \times & 0 & \times & 0 \\
0 & 0 & 0 & 0 & \times & 0 & 0 & \times \\
0 & 0 & 0 & 0 & 0 & \times & \times & 0 \\
0 & 0 & 0 & 0 & 0 & \times & 0 & \times \\
\times & 0 & 0 & 0 & \times & 0 & 0 & 0 \\
\times & 0 & 0 & 0 & 0 & \times & 0 & 0 \\
\times & 0 & 0 & 0 & 0 & 0 & \times & 0 \\
\times & 0 & 0 & 0 & 0 & 0 & 0 & \times \\
0 & \times & 0 & 0 & \times & 0 & 0 & 0 \\
0 & \times & 0 & 0 & 0 & \times & 0 & 0 \\
0 & \times & 0 & 0 & 0 & 0 & \times & 0 \\
0 & \times & 0 & 0 & 0 & 0 & 0 & \times \\
0 & 0 & \times & 0 & \times & 0 & 0 & 0 \\
0 & 0 & \times & 0 & 0 & \times & 0 & 0 \\
0 & 0 & \times & 0 & 0 & 0 & \times & 0 \\
0 & 0 & \times & 0 & 0 & 0 & 0 & \times \\
0 & 0 & 0 & \times & \times & 0 & 0 & 0 \\
0 & 0 & 0 & \times & 0 & \times & 0 & 0 \\
0 & 0 & 0 & \times & 0 & 0 & \times & 0 \\
0 & 0 & 0 & \times & 0 & 0 & 0 & \times
\end{pmatrix}
$$

Fig. 5. Sparsity structure of log-example on 8 columns.

coloring algorithms is observed with Harwell-Boeing problems. Neither DSATUR nor Small-k could solve the abb313GPIA problem (with row partition Π_m) after being run for 3 days on a Sun Blade 10 running Solaris. Chaff (a SAT solver) together with efficient encoding was able to solve this problem (chromatic number 9). Small-k did solve the remaining five problems from the test suite while DSATUR did not return on ash958GPIA with $k = m$ row partition. Small-k found the optimal coloring (4-coloring) using a total of 11.68 seconds. Unlike the Eisenstat and log-examples, however, the optimal coloring (4-coloring) of ash958GPIA was obtained with $k = m$ row partition.

Table 1
DSATUR Coloring of Eisenstat Graphs.

n	Partition (k)	Nodes		Edges		Lower bound		Subproblems		Best coloring	
		u	p	u	p	u	p	u	p	u	p
6	1	6	6	15	15	6	6	1	1	6	6
	3	15	13	60	50	4	6	12	8	5	6
	4	16	15	65	60	4	5	13	11	5	5
	7	18	18	75	75	3	3	40	36	4	4
8	1	8	8	28	28	8	8	1	1	8	8
	3	20	17	121	97	5	7	30	11	6	8
	5	25	24	168	158	5	6	21	19	6	6
	9	28	28	198	198	5	5	39	50	6	6
16	1	16	16	120	120	16	16	1	1	16	16
	3	40	33	543	410	9	16	54	18	10	16
	4	48	43	756	664	10*	12*	80	66	11	13
	9	69	67	1494	1414	9*	8*	61	375	10	11
	17	88	88	2460	2460	8*	8*	153	184	10	10

Table 2
Exact coloring of SMTAPE collection using coloring routines: DSATUR, Chaff, and Small-k.

Matrix	$G_\Pi(A),\ q=1$			$G_\Pi(A),\ q=m$		
	Nodes	*Edges*	$\chi(G_\Pi(A))$	*Nodes*	*Edges*	$\chi(G_\Pi(A))$
ash219	85	219	4	438	2205	4
abb313	176	3206	10	1557	65390	(10)9
ash331	104	331	6	662	4185	4
will199	199	960	7	701	7065	7
ash608	188	608	6	1216	7844	4
ash958	292	958	6	1916	12506	(5)4

6 Concluding Remarks

This paper describes an application of graph coloring ideas in the numerical determination of large sparse derivative matrices. Other related work can be found in [2,4,5,15]. We note that the procedure for the determination of Jacobian matrices as outlined in Section 2 can be given in a more general way [16]. Our graph generator

provides an object-oriented implementation of the column segments approach for efficient determination of sparse Jacobian matrices. As well, column segment graph generator provides a convenient tool for obtaining computationally hard coloring instances. The computational test results indicate that on majority of real-world test problem of moderate size, exact coloring can be combined with other heuristics to make it viable by, for example, terminating the coloring procedure as soon as an "acceptable" coloring has been obtained.

There are a number of research directions for further investigation related with this work. The current graph generator is based on one-directional partitioning. Can this procedure be generalized to two-directional partitioning [5,15]? The column segment coloring is what is known as distance-1 coloring. A natural extension to this work is to generalize it to distance-k coloring [2,4,19] instance generators.

Acknowledgement: The work of the first author was supported by the Natural Sciences and Engineering Research Council of Canada under RGPIN and the work of the second author was supported by the Research Council of Norway. The first author thanks his research assistant Zhenshuan Zhang for help with the numerical experiments.

References

[1] D. Brélaz. New methods to color the vertices of a graph. *Commun. ACM*, 22(4):251–256, Apr. 1979.

[2] T. F. Coleman and J.-Y. Cai. The cyclic coloring problem and estimation of sparse Hessian matrices. *SIAM J. Alg. Disc. Meth.*, 7(2):221–235, 1986.

[3] T. F. Coleman and J. J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM J. Numer. Anal.*, 20(1):187–209, 1983.

[4] T. F. Coleman and J. J. Moré. Estimation of sparse Hessian matrices and graph coloring problems. *Math. Programming*, 28:243–270, 1984.

[5] T. F. Coleman and A. Verma. The efficient computation of sparse Jacobian matrices using automatic differentiation. *SIAM J. Sci. Comput.*, 19(4):1210–1233, 1998.

[6] J. Culberson and I. Gent. Frozen development in graph coloring. *Theoretical Computer Science*, 265(1–2):227–264, 2001.

[7] A. R. Curtis, M. J. D. Powell, and J. K. Reid. On the estimation of sparse Jacobian matrices. *J. Inst. Math. Appl.*, 13:117–119, 1974.

[8] J. Dongarra, A. Lumsdaine, R. Pozo, and K. Remington. A sparse matrix library in c++ for high performance architectures. In *Proceedings of the Second Object Oriented Numerics Conference*, pages 214–218, 1994.

[9] I. S. Duff, R. G. Grimes, and J. G. Lewis. Users' guide for the Harwell-Boeing sparse matrix collection (Release I). Technical Report tr/pa/92/86, CERFACS, 1992.

[10] A. V. Gelder. Another look at graph coloring via propositional satisfiability, 2002. Computational Symposium on Graph Coloring and Generalizations (COLOR02), Ithaca, NY, 7-8 September 2002.

[11] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[12] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, Penn., 2000.

[13] A. Griewank and C. Mitev. Detecting Jacobian sparsity patterns by Bayesian probing. *Math. Prog.*, 93(1):1–25, 2002.

[14] A. Griewank and P. L. Toint. On the unconstrained optimization of partially separable functions. In M. J. D. Powell, editor, *Nonlinear Optimization 1981*, pages 301–312. Academic Press, New York, 1982.

[15] A. S. Hossain and T. Steihaug. Computing a sparse Jacobian matrix by rows and columns. *Optimization Methods and Software*, 10:33–48, 1998.

[16] S. Hossain and T. Steihaug. Reducing the number of AD passes for computing a sparse Jacobian matrix. In G. Corliss, C. Faure, A. Griewank, L. Hascoët, and U. Naumann, editors, *Automatic Differentiation: From Simulation to Optimization*, Computer and Information Science, chapter 31, pages 263–270. Springer, New York, 2001.

[17] S. Hossain and T. Steihaug. Optimal Direct Determination of Sparse Jacobian Matrices. Technical Report 254, Department of Informatics, University of Bergen, Norway, October 2003.

[18] S. Hossain and J. Zhang. CsegGraph: Column Segment Graph Generator. Technical Report, Department of Mathematics and Computer Science, University of Lethbridge, 4401 University Drive, Alberta, Canada, 2003.

[19] S. T. McCormick. Optimal approximation of sparse Hessians and its equivalence to a graph coloring problem. *Math. Programming*, 26:153–171, 1983.

[20] J. J. Moré and J. Wu. Optimal sizing for a class of nonlinearly elastic materials. *SIAM J. Optim.*, 7(3):814–836, 1997.

[21] G. N. Newsam and J. D. Ramsdell. Estimation of sparse Jacobian matrices. *SIAM J. Alg. Disc. Meth.*, 4(3):404–417, 1983.

[22] L. B. Rall and T. W. Reps. Algorithmic differencing. In A. Facius and U. Kulisch, editors, *Perspectives on Enclosure Methods*. Springer-Verlag, Vienna, 2001.