The Cyclic Coloring Problem and Estimation of Sparse Hessian Matrices

Thomas F. Coleman*
Jin-yi Cai

TR 84-646
October 1984

Department of Computer Science
Cornell University
Ithaca, New York 14853

* The work of this author was supported in part by the Applied Mathematical Scienced Research Program (KC-04-02) of the Officed of Energy Research of the U.S. Department of Energy under Contract DE-AC02-83ER13069.

THE CYCLIC COLORING PROBLEM

AND

ESTIMATION OF SPARSE HESSIAN MATRICES

Thomas F. Coleman *
Jin-yi Cai

Computer Science Department
Cornell University
Ithaca, New York 14853

October 1984

Abstract

Numerical optimization algorithms often require the (symmetric) matrix of second derivatives, $\nabla^2 f(x)$, of some problem function $f: R^n \rightarrow R^1$. If the Hessian matrix is large and sparse then estimation by finite differences can be quite attractive since several schemes allow for estimation in $<< n$ gradient evaluations.

The purpose of this paper is to analyze, from a combinatorial point of view, a class of methods known as substitution methods. We present a concise characterization of such methods in graph-theoretic terms. Using this characterization, we develop a complexity analysis of the general problem and derive a roundoff error bound on the Hessian approximation. Moreover, the graph model immediately reveals procedures to effect the substitution process optimally (ie. using fewest possible substitutions given the differencing directions) in space proportional to the number of nonzeroes in the Hessian matrix.

* The work of this author was supported in part by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under Contract DE-AC02-83ER13069

1. Introduction

We are concerned with the estimation of a large sparse symmetric matrix of second derivatives $\nabla^2 f(x)$ for some problem function $f: R^n \rightarrow R^1$. In particular, we note that the product $\nabla^2 f(x) \cdot d$ can be estimated, for example, by forward differences

$$\nabla^2 f(x) \cdot d = [\nabla f(x+d) - \nabla f(x)] + o(||d||) \tag{1.1}$$

When the structure of $\nabla^2 f(x)$ is known then usually a few well-chosen differencing directions $d_1, \ldots, d_p$ affords the recovery of estimates of all nonzeroes of $\nabla^2 f(x)$. Let us denote our estimate by H. We will assume that the sparsity pattern of H is known; the diagonal elements are specified as nonzero; H is symmetric.

There has been considerable work recently concerned with this problem, especially with trying to make p as small as possible. Curtis, Powell, and Reid [1974] suggested a method, *CPR*, for the unsymmetric problem. Their idea was to build groups of independent columns in a left-to-right greedy fashion. (2 vectors x, y are ***independent*** if $x_i \neq 0 \Rightarrow y_i = 0$.) It is easy to see that such a p-partition allows for the estimation of a matrix with p differencing directions. Specifically, let $C_1, \ldots, C_p$ be a partition of the columns of H where each group consists of independent columns. Let h_i be the steplength associated with column i, $i=1,\ldots,n$, and define

$$d_k = \sum_{i \in S_k} h_i e_i$$

where S_k is the set of indices to columns in group C_k, and e_i is the i^{th} column of the identity matrix. Then, if $[\nabla f(x+d_k) - \nabla f(x)]_i \neq 0$ it follows that there is exactly one column j in group C_k with H_{ij} a designated nonzero and we can assign

$$H_{ij} \leftarrow \frac{[\nabla f(x+d_k) - \nabla f(x)]_i}{h_j}$$

Coleman and Moré [1983] analyzed and modified this method by taking a combinatorial point of view. In particular, a *column intersection graph* can be formed by associating with each column i of H a node v_i and defining an edge between node v_i and node v_j iff there is an index k such that both H_{ki} and H_{kj} are nonzeroes. A p-coloring of this graph is an assignment, ϕ, of 'colors' to nodes such that if there is an edge between node v_i and node v_j then $\phi(v_i) \neq \phi(v_j)$. It is not hard to see that a p-coloring of this graph induces a valid partition of independent columns and vice versa.

Coleman, Garbow, and Moré [1984a] have developed FORTAN 77 codes based on this work. Such (unsymmetric) methods can be applied to the symmetric problem (McCormick [1983] discusses the complexity of this approach) however it is probably worthwhile using symmetry when it is present.

Powell and Toint [1979] were the first to exploit symmetry. They pointed out that symmetry can be used both in a direct and an indirect fashion. A direct method is one in which each unknown of H is determined independently of the others. More specifically, let $C_1, \ldots, C_p$ be a partition of the columns of H. Since each off-diagonal nonzero is represented twice in H it is no longer necessary that each group consist of independent columns. It is necessary, however, that for each nonzero (i,j) **either** column i reside in a group C_r such that no other column in this group has a nonzero in row j **or** column j reside in a group C_s such that no other column in this group has a nonzero in row i. If the latter condition were true then H_{ij} would be determined

$$H_{ij} \leftarrow \frac{[\nabla f(x+d_s)-\nabla f(x)]_i}{h_j}$$

and $H_{ji} \leftarrow H_{ij}$. Clearly a similar (symmetric) rule would hold for the former condition.

Coleman and Moré [1984] analyzed such methods from a combinatorial point of view and produced a simple graph-theoretic characterization of all partitions that can be used to induce a direct symmetry-exploiting determination of H. Let us represent the structure of H by the usual adjacency graph $G(H) = (V(H), E(H))$. That is, if H is a symmetric matrix of order n then $V(H)$ consists of n vertices $v_1, \ldots, v_n$ (associate column i of H with vertex v_i) and $E(H)$ consists of pairs of vertices (edges) where $(v_i, v_j) \in E(H)$ if and only if H_{ij} (H_{ji}) is considered a nonzero. A p-partition of the columns of H, $C_1, \ldots, C_p$ can be viewed as an assignment of colors , ϕ, to the nodes of G, ϕ: $V \rightarrow \{1, \ldots, p\}$. This assignment is a *$p$-coloring* if $(v,w) \in E \Rightarrow \phi(v) \neq \phi(w)$. A *path p-coloring* is a p-coloring with the additional stipulation that every path in G of length 4 (distinct) vertices uses at least 3 colors. The characterization of direct symmetric methods given by Coleman and Moré is simply

Theorem 1.1: The mapping ϕ is a path p-coloring if and only if ϕ induces a partition of the columns of H consistent with direct determination.

Note: In order to avoid confusion in this paper we have changed the notation used by Coleman and Moré [1984] - here we use 'path p-coloring' instead of 'symmetric p-coloring'.

This characterization led to a deeper understanding of the direct estimation problem on symmetric structures which in turn yielded a complexity analysis and algorithmic possibilities.

Indirect estimation of symmetric matrices may be preferable because fewer groups (ie. differencing directions) will be needed, in general. Powell and Toint concentrated on substitution methods where directions are chosen so that nonzeroes can be determined via a substitution process. (They restricted their attention, as we do, to substitution methods based on a partition of columns.) So in this case there is interdependence of the matrix unknowns (nonzeroes) to the degree that an underlying lower triangular system is defined. Powell and Toint proposed an algorithm to determine the differencing directions and then solve for the unknowns (lower triangular substitution method (LTS)). Subsequently, Coleman and Moré [1984] analyzed this process from a combinatorial point of view. This analysis led to a modified and emperically superior procedure (the resulting FORTAN 77 code is described in Coleman, Garbow, and Moré [1984b]). However, a simple insightful characterization, in the vein of Theorem 1.1, was not provided.

The purpose of this paper is to provide such a characterization. This result is as simple as Theorem 1.1 and is clearly the analagous result. This view provides enormous insight into the combinatorial nature of the problem as well as suggesting algorithmic possibilities. Furthermore, the graph theoretic interpretation reveals that if a partition of columns allows for the recovery of H via a substitution process then it is always possible to do so efficiently. In particular, every unknown can be solved for in (roughly) less than $n/2$ substitutions and the space required to compute H is proportional to the number of nonzeroes. This is somewhat surprising since the Powell-Toint procedure relies heavily on a regular matrix structure produced by LTS which is not present for an arbitrary feasible partition. Finally, the graph model allows one to derive a growth of error bound for a general substitution method, which is essentially analagous to the result achieved by Powell and Toint for a specific method, LTS.

Section 2 will provide the characterization of substitution methods followed by a roundoff error discussion. In Section 3 we establish the complexity of the problem and discuss its combinatorial relationship to the symmetric direct problem (path coloring). Section 4 deals with algorithms for effecting the substitution process in space proportional to $|E|$ (i.e. the number of nonzeroes of H). Finally, observations on parallelism are provided in Section 5.

2. Substitution Methods and Cyclic Coloring

A partition of columns of a symmetric matrix induces a substitution method if there is an ordering of the matrix unknowns such that all unknowns can be solved for, in that order, using symmetry and previously solved elements. This notion is fully general (subject to the partition restriction) but seems to be a difficult one to work with. There is however a very elegant and simple graph theoretic interpretation. The major purpose of this section is to present this characterization. We conclude this section by applying this characterization to achieve bounds on the maximum number of substitutions and hence we bound the potential growth of roundoff error.

First it is necessary to formalize the concept of a substitution method in matrix terms. Let U be the set of indices of matrix unknowns (identify (i,j) with (j,i)) and suppose that U is ordered: $U = \{(i_k, j_k)\}$. Let the columns of H be partitioned $\{C_1, \ldots, C_p\}$ and define

$$S_0 = \emptyset; \quad S_k = S_{k-1} \bigcup \{(i_k, j_k)\}, \quad 1 \leq k \leq |U|.$$

The ordering *induces a substitution method* iff

either j_k belongs to a group C, say, and if l is any other column in C with a nonzero in row i_k then $(i_k, l) \in S_{k-1}$ **or** i_k belongs to a group C', say, and if l' is another column in C' with a nonzero in row j_k then $(j_k, l') \in S_{k-1}$.

The essence of this statement is that, at the k^{th} step, it is possible to solve for element (i_k, j_k) or, equivalently (j_k, i_k), by substitution. We call a partition, for which there exists such an ordering, *substitutable*. For example, if H is a tridiagonal matrix then it is easy to verify that the partition ({1,3,...} ,{2,4,...}) is substitutable.

Obviously there are substitutable partitions for any symmetric matrix. For example, every partition consistent with a path coloring is substitutable. Alternatively, a partition that induces a 'lower triangular substitution method' is substitutable. (Coleman and Moré [1984] and Powell and Toint [1979] discussed lower triangular substitution methods.) However, here we are interested in minimizing the number of groups in a general substitutable partition. The above 2 examples are restrictive in that they consider only particular classes of substitutable partitions. The general problem is

Partition Problem: Obtain a substitutable partition of the columns of a given symmetric matrix H with the fewest groups.

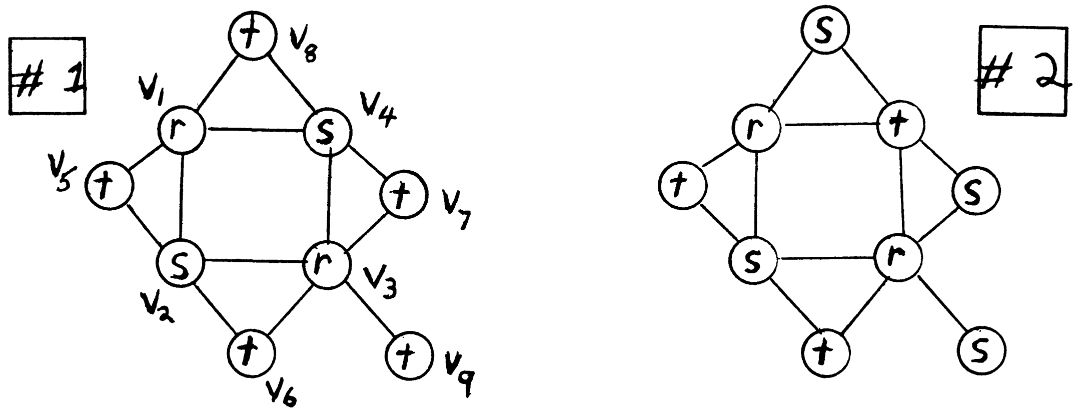
How difficult is the partition problem ? This is a hard question to answer considering the rather clumsy matrix formalization of a substitution method. Fortunately a substitutable partition has a simple expression in the language of graphs.

Definition. A mapping $\phi : V \rightarrow \{1,2,...,p\}$ is a *cyclic p-coloring* of G if ϕ is a p-coloring and if ϕ uses at least 3 colors in every cycle of G.

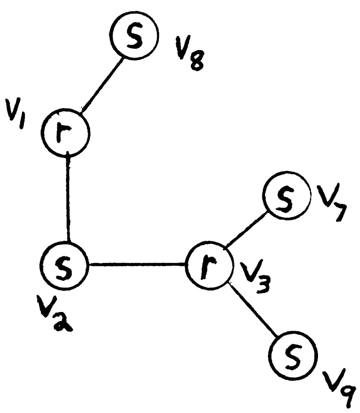
As the following theorem indicates, we now have a simple characterization of a substitutable partition.

Theorem 2.1. Let H be a symmetric matrix with a nonzero diagonal. The mapping ϕ induces a substitution method $\Leftrightarrow$ ϕ is a cyclic coloring of $G(H)$.

Before providing the proof, let us consider an informal argument based on the following example. Let the adjacency graph of H, $G(H)$ be as shown.



Both assignments of the colors r,s,t are valid colorings but assignment 1 is **not a valid cyclic coloring**: the cycle v_1,v_2,v_3,v_4 uses only 2 colors. Assignment 2 is a valid cyclic coloring. The edges (off-diagonal nonzeroes) can be determined by considering each pair of colors in turn. For example, consider the subgraph, $F_{r,s}$, induced by the nodes colored r or s:



Edges (1,8), (3,7), and (3,9) can all be determined immediately. Consider for example edge (3,7). Column 3 has a a nonzero in row 7 and resides in group C_r. There is no other column in group C_r with a nonzero in row 7 (else node 7 would have another incident r-node). Therefore, $H_{7,3}$ (hence $H_{3,7}$) can be determined directly. Once (3,7) and (3,9) are determined, edge (2,3) can be computed: column 2 has a nonzero in row 3 and resides in group C_s. Columns 7 and 9 are the other columns in group C_s with nonzeroes in row 3. However, $H_{3,7}$ and $H_{3,9}$ are now known quantities; hence, $(H_{3,2})$ can be computed with 2 substitutions.

Clearly the process can be carried to completion until every edge in $F_{r,s}$ is determined. (It is trivial to see that the diagonal elements can be directly determined: this follows from the fact that ϕ is a coloring .) But every pair of colors induces a forest, otherwise ϕ would not be a cyclic coloring, and therefore every nonzero can be determind by considering each pair of colors in turn.

The 'if' part of Theorem 2.1 is proved along the lines of the example given above. The 'only if' part is perhaps a bit surprising but not difficult to prove.

Proof of Theorem 2.1. First we prove that every cyclic coloring of $G(H)$ induces a substitution method. Since ϕ is a coloring, every diagonal element of H can be computed. Consider next $(i,j) \in U$, $i \neq j$. Suppose that column i is in group C_r and column j is in group C_s. Clearly, since ϕ is a coloring, $r \neq s$. Consider the subgraph induced by the nodes colored r and the nodes colored s, say $F_{r,s}$. Since ϕ is a cyclic coloring, $F_{r,s}$ contains no cycle and therefore is a forest. The edges in $F_{r,s}$ correspond to off-diagonal unknowns of H. They can be solved, or ordered, independent of the rest of the unknowns of H. In particular, each leaf-incident edge can be solved directly since there is no conflict. We can now 'delete' all such edges and consider each new leaf-incident edge. Each such edge is now incident to known edges and can therefore be solved. Clearly the process can be repeated until an edge-less graph remains. The entire procedure can now be repeated for each pair of colors until every unknown is determined.

We now show that if ϕ induces a substitution method then ϕ is a cyclic coloring. First it is clear that ϕ must be a valid coloring, otherwise the diagonal elements would not be determined. Suppose then that ϕ is a coloring but is not a cyclic coloring. Hence there must be a cycle, with at least 4 edges, colored with just 2 colors, say r, s. Let (i,j) be the **first** edge in this cycle to be solved (ordered) and let us assume, without loss of generality, that v_i is colored r and v_j is colored s. Let node v_i be incident also to node v_h (on the cycle) and let v_j be incident also to node v_k (on the cycle). But (i,j) cannot be determined from group C_s because columns j and h both reside in this group with nonzeroes in row i (and (i,h) is not yet known (ordered)). Similarily, (j,i) is not determined from group C_r because columns i and k both reside in this group with nonzeroes in row j (and (j,k) is not yet known (ordered)). Therefore no edge in this cycle can be solved first and ϕ cannot induce a substitution method. •

Hence the partition problem is equivalent to the
Cyclic Coloring Problem: Obtain a minimum cyclic coloring of $G(H)$.
Note that once we have found a cyclic coloring of $G(H)$ then the coloring induces a substitutable partition and the corresponding ordering of U is available, as the proof of Theorem 2.1 indicates. A tridiagonal matrix provides a trivial example. The graph is just

$G(H) =$ 

and a valid cyclic coloring is provided by assigning r to the even nodes and s to the odd nodes. The diagonal elements can be solved directly and the off-diagonal elements are obtained via substitution: edges (1,2) and $(n-1,n)$ are obtained first (directly), followed by (2,3) and $(n-2,n-1)$, with 1 substitution each, and so on. The middle node will be the last determined element with approximately ½ n dependencies or substitutions.

Suppose we modify the above example by adding an edge from node v_1 to node v_n. A cyclic coloring would then require 3 colors; for example, we could use our previous assignment except we apply a new color, t, to node 1. Now (1,2) and $(1,n)$ can be determined directly (or, ordered first) and the remaining elements can be determined, as before, via substitution.

The above 2 examples raise an interesting question with numerical significance: Is there a limit to the number of dependencies or substitutions? The potential growth of roundoff error as well as the amount of computational work will depend on this number and therefore a bound, tighter than the total number of nonzeroes in H, would be consequential. Powell and Toint[1979] established that for a particular class of substitution methods, triangular substitution methods, the bound is $n-2$. The cyclic coloring characterization leads us immediately to a more general result. Every unknown can be determined by considering the forest induced by a particular pair of colors. But each forest can have at most $n-1$ edges and therefore we have the following result.

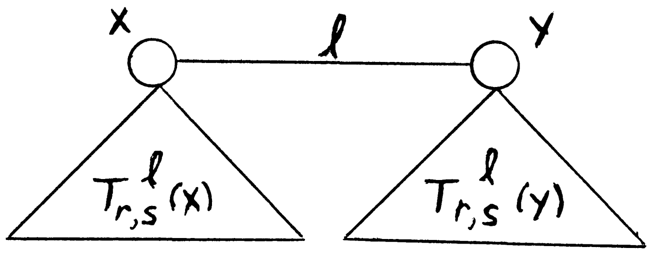
Theorem 2.2. Let ϕ be a substitutable partition. Then, each unknown in H is dependent on at most $n-2$ other unknowns. $\bullet$

Clearly this result is the best possible worst case upper bound, if we allow any possible feasible ordering of the unknowns or edges. To see this just consider the tridiagonal case: if the edges are solved from one end of $G(H)$ to the other, then the last edge requires $n-2$ substitutions. However, certain orderings are preferable over others. For example, in the tridiagonal case one can achieve a bound of $\lfloor \frac{1}{2}\,(n-2) \rfloor$ if each edge is solved by substituting from the nearest end of $G(H)$. It is not hard to see that, over different orderings, this is the best possible worst case upper bound; again, just consider the tridiagonal case.

Is it possible to order the unknowns, in general, so that the maximum number of substitutions is less than or equal to $\lfloor \frac{1}{2}\,(n-2) \rfloor$? In order to answer this question, consider when it is feasible, during the solution process, to solve for edge $l \triangleq (x,y)$ in $T_{r,s}$ where $T_{r,s}$ is a tree in the forest induced by the colors r and s. Define

$$T^l_{r,s}(x) = (V^l_{r,s}(x), E^l_{r,s}(x)) \quad T^l_{r,s}(y) = (V^l_{r,s}(y), E^l_{r,s}(y))$$

to be the 2 subtrees, rooted at x and y respectively, that remain when edge l (but not its endpoints) is removed from $T_{r,s}$. That is

$T_{r,s} =$ 

It is clear that (x,y) is ready to be solved **if and only if either** every edge in $T^l_{r,s}(x)$ is solved **or** every edge in $T^l_{r,s}(y)$ is solved. Furthermore, (x,y) requires at least

$$mincost(x,y) \triangleq \min\{\, |E^l_{r,s}(x)|,\ |E^l_{r,s}(y)| \,\}$$

substitutions. Note that an ordering that computes edge (x,y) using $mincost(x,y)$ substitutions, for each edge (x,y), is optimal and requires less than $\lfloor \frac{1}{2}\,(n-2) \rfloor$ substitutions for each edge.

Such an ordering is possible and is provided by the following algorithm. Let $T \triangleq (V_T, E_T)$ be the tree under consideration, with $|V| = n_T \leq n$.

```
Algorithm solve_tree
  T_1 ≜ (V_1,E_1) where V_1 = V_T, E_1 = E_T
  for each vertex v ∈ V_1 do value(v) ← 0 od
  for i=1 to |E_1| do
    choose a leaf x_i of T_i, of smallest value
    let y_i be the vtx such that (x_i,y_i) ∈ T_i
    value(y_i) ← value(y_i) + value(x_i) + 1
    solve (x_i,y_i)
    E_{i+1} ← E_i - {(x_i,y_i)}
    V_{i+1} ← V_i - {x_i}
    T_{i+1} ← (V_{i+1},E_{i+1})
  od
end solve_tree
```

Theorem 2.3: Algorithm *solve_tree* solves for each edge $(x,y) \in E_T$ using the fewest possible substitutions, $mincost(x,y)$. Hence, *solve_tree* requires at most

$$\max\{mincost(x,y): (x,y) \in E\} \leq \left\lfloor \tfrac{1}{2} (n_T - 2) \right\rfloor \leq \lfloor \tfrac{1}{2} (n-2) \rfloor$$

substitutions to determine any edge of E_T.

Proof: Assume then that *solve_tree* doesn't solve for each edge using the fewest possible substitutions. First we establish that algorithm *solve_tree* terminates: Since T_i is a tree and x_i is a leaf in T_i it follows that T_{i+1} is a tree. Therefore T_{i+1} will have a leaf (indeed at least 2) and x_{i+1} will be found. It follows that the algorithm will determine every edge.

Next, let us assume that at step i vertex x_i is the chosen leaf in T_i and edge $l \triangleq (x_i,y_i)$ is determined non-optimally. That is, $|E^l(x_i)| > |E^l(y_i)|$ and hence $|E^l(x_i)| = value(x_i) > \left\lfloor \tfrac{1}{2} (n_T-2) \right\rfloor$. Consider a leaf, v_i, in $T_i \bigcap T^l(y_i)$ (there must be at least 1). But

$$|E^l(x_i)| > \left\lfloor \tfrac{1}{2} (n_T-2) \right\rfloor \Rightarrow |E^l(y_i)| \leq \left\lfloor \tfrac{1}{2} (n_T-2) \right\rfloor$$

and therefore,

$$value(v_i) \leq \left\lfloor \tfrac{1}{2} (n_T-2) \right\rfloor < value(x_i)$$

Therefore (x_i, y_i) would not be solved at step i, a contradiction. •

In summary, Theorem 2.3 says that for an arbitrary substitutable partition algorithm *solve_tree* will compute each edge with the fewest possible substitutions (with respect to that partition) and that number is always bounded by $\lfloor \frac{1}{2}(n-2) \rfloor$.

We conclude Section 2 by considering the accuracy of the estimated Hessian matrix in more detail. We will show that an error bound, similar to that achieved by Powell and Toint [1979] for a particular class of algorithms (lower triangular substitution methods) holds for any substitution method provided the unknowns are solved for in the manner suggested by *solve_tree*.

Every substitutable partition with p groups, or cyclic p-coloring, allows for the recovery of the matrix unknowns via a back substitution process provided the differencing vectors are consistent with the coloring ϕ. In particular, let S_k denote the set of nodes (columns) colored k (ie. in C_k) and define

$$d_k = \sum_{i \in S_k} h_i e_i$$

where h_i is the step-length associated with column i. Let x be a given point in R^n and define $u_k = \nabla f(x+d_k) - \nabla f(x)$, for $k = 1,\ldots,p$. If H denotes the approximation to $\nabla^2 f(x)$, it follows that, since ϕ is a coloring,

$$H_{jj} \cdot h_j = [\nabla f(x + h_j e_j)]_j - [\nabla f(x)]_j, \quad j = 1,\ldots,n$$

The diagonal approximations are defined by these equations, and will not participate in any subsequent calculations. Indeed such equations usually guide the choice of h_j: h_j is chosen to balance truncation and roundoff errors in order to approximate the diagonal elements as accurately as possible (eg. Gill, Murray, Saunders, and Wright [1983].)

Previous analysis has shown that it is only necessary to consider 2 colors (directions) at a time when solving for the off-diagonal elements. Let us concern ourselves then with a tree, $T_{r,s}$, induced by colors r and s. Let $u_r = \nabla f(x+d_r) - \nabla f(x)$, $u_s = \nabla f(x+d_s) - \nabla f(x)$ and let

$$\hat{u}_r = u_r + \epsilon_r, \quad \hat{u}_s = u_s + \epsilon_s$$

denote the computed quantities (ie. contaminated with rounding error).

The solution process is provided by algorithm *solve_tree* with the statement ' *solve* (x_i, y_i) ' expanded, to read

$$H_{ij} \cdot h_j = (\hat{u}_c)_i - \sum_{k \in N(i)} H_{ik} \cdot h_k \tag{2.1}$$

where we **identify vertex x_i with index i, and vertex y_i with index j.** $N(i)$ is the set of neighbours of node x_i in $T^l_{r,s}(x_i)$, and $c = \phi(y_i)$ (which is one of r,s). In other words, when H_{ij} is solved for, every other element in row i of columns in group C_c has already been solved for; the right-hand-side of (2.1) is adjusted accordingly.

Following Powell and Toint, we define the error matrix F to be $H - \nabla^2 f$ and let

$$(\delta_c)_i = (\hat{u}_c)_i - \sum_{k \in N(i) \bigcup \{j\}} (\nabla^2 f(x))_{ik} \cdot h_k \tag{2.2}$$

In words, $(\delta_c)_i$ measures the difference between the computed quantity $(\hat{u}_c)_i$ and the ideal $(\nabla^2 f(x) \cdot d_c)_i$. Hence $(\delta_c)_i$ is a composite of roundoff and truncation errors. If we assume that the second derivatives of f are Lipschitz continuous then a standard bound is obtained:

$$M \triangleq \max_{c,i} \{ |(\delta_c)_i| \} \leq C \cdot \max_k \{ |h_k|^2 \} + \max_{c,i} \{ |(\epsilon_c)_i| \}$$

where C is a positive constant.

The following result establishes a bound on the elements in the error matrx F.

Theorem 2.4: If H is obtained by algorithm *solve_tree* (with 'solve (x_i, y_i)' effected by 2.1) then

$$|F_{ij}| \leq (|E^l_{r,s}(x_i)| + 1) \cdot M \cdot \max_{i,j,k} \{ \frac{|h_k|}{|h_i h_j|} \}$$

$$\leq (\lfloor \tfrac{1}{2}(n-2) \rfloor + 1) \cdot M \cdot \max_{i,j,k} \{ \frac{|h_k|}{|h_i h_j|} \}$$

where again we identify column i with node x_i, column j with node y_i, $l \triangleq (x_i, y_i)$, and $\{\phi(x_i), \phi(y_i)\} = \{r,s\}$.

Proof: Combining (2.1), (2.2) and the definition of F yields

$$(\delta_c)_i = F_{ij} \cdot h_j + \sum_{k \in N(i)} F_{ik} \cdot h_k$$

which implies the bound

$$|F_{ij} h_i h_j| \leq |(\delta_c)_i \cdot h_i| + \sum_{k \in N(i)} |h_i h_k F_{ik}|$$

$$= |h_i (\delta_c)_i| + \sum_{k \in N(i)} |F_{ki} h_k h_i|$$

But this same decomposition can be applied, recursively, to each $F_{ki} h_k h_i$, for $k \in N(i)$, to yield

$$|F_{ij} h_i h_j| \leq \sum_{k \in V_{r,s}^l(x_i)} |(\delta_c)_k \cdot h_k| \tag{2.3}$$

Since $V_{r,s}^l(x_i) = |E_{r,s}^l(x_i)| + 1$, the result follows immediately from (2.3) and Theorem 2.3. •

One can conclude from this result that the growth of roundoff error is reasonably limited if the steplength does not vary greatly in size. On the other hand, if there is significant variance (recall that stepsizes are chosen to accurately approximate diagonal elements) then this result may allow for unacceptable growth of error: a direct method may be preferable under these circumstances.

3. The Cyclic Chromatic Number

How difficult is the cyclic coloring problem? We address this question in this section. The reader who is unfamiliar with the fundamentals of complexity theory and *NP*-completeness is urged to consult the excellent resource book "Computers and Intractability: A Guide to the Theory of *NP*-Completeness", by Michael R. Garey and David S. Johnson [1979].

We will first consider the cyclic coloring decision problem (*CCDP*) and show that this problem is *NP*-complete: we do this by transforming the general graph coloring decision problem (*CDP*). We then conclude that the corresponding optimization problem, the cyclic coloring problem, is *NP*-hard. The consequence of this result is just this: if we could solve the cyclic coloring problem in polynomial time (*P*-time) then we could also solve the graph coloring problem in *P*-time (as well as a host of other 'intractable' problems.) Since this is deemed highly unlikely, an expedient approach to our problem is to investigate efficient heuristic and approximation schemes (we discuss this in Section 4).

It is common, when considering complexity questions related to discrete optimization problems, to consider the decision problem formulation. In this case we have the

Cyclic Coloring Decision Problem (CCDP): Given an integer $p \geq 3$ and an arbitrary graph G, is it possible to assign a cyclic p-coloring to the nodes of G?

We have excluded the simple cases $p = 1,2$ since it is easy to see that polynomial algorithms exist for such cases. The following theorem shows that *CCDP* is not so simple for $p \geq 3$.

Theorem 3.1: *CCDP* is *NP*-complete.

Proof: The first step is to show that CCDP is in the class *NP*. In particular, we must show that we can validate, in *P*-time, whether or not a particular assignment of p colors is indeed a cyclic coloring. To do this one must merely consider each pair of colors, in turn, and decide whether or not the induced graph is a forest. Clearly this is a polynomial time operation.

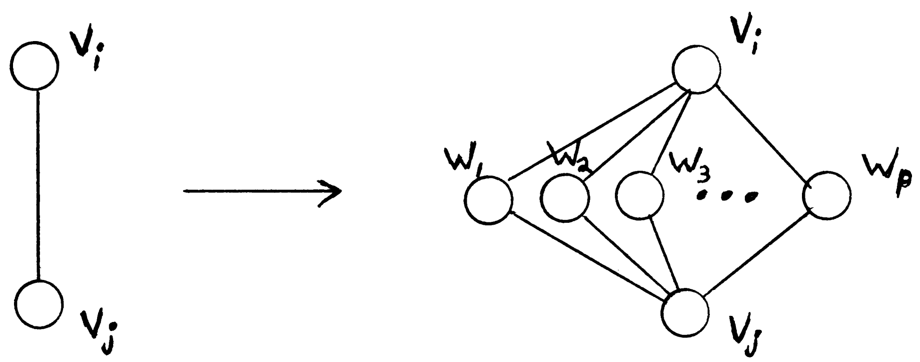
We now proceed to transform the general coloring problem (CDP), which is is known to be *NP*-complete, to *CCDP*. Consider an arbitrary graph $G = (V,E)$ and integer $p \geq 3$. Let $|V| = n$ and $|E| = m$. We construct a new graph, $G' = (V',E')$ as follows. For each edge $e_l = (v_i,v_j) \in E$, define a bipartite graph G_l' with vertices

$$\{ v_i, v_j, w_1^{(l)}, \ldots, w_p^{(l)} \}$$

and edges

$$(v_i, w_k^{(l)}), \quad (v_j, w_k^{(l)}), \quad k = 1,\ldots,p \; .$$

Graphically, this transformation is



Now define a bipartite graph G' by setting

$$V' = V(G) \bigcup \{ w_k^{(l)} : 1 \leq k \leq p, \quad 1 \leq l \leq m \}$$

and

$$E' = \bigcup_{l=1}^{m} E(G_l') .$$

We now show that if G can be p-colored using an assignment ϕ then G' can be assigned a cyclic p-coloring, ϕ'. In particular, for each $v \in V$ let $\phi'(v) = \phi(v)$. Hence if we consider any G_l', induced by $e_l = (v_i, v_j) \in E$, then $\phi'(v_i) \neq \phi'(v_j)$. Let ϕ' assign vertices w_j^l, $j=1,...,p$ any color different from $\phi'(v_i)$ and $\phi'(v_j)$.

We claim that ϕ' is a cyclic p-coloring of G': Clearly any cycle in G' must contain a path $(v_i, w_k^{(l)}, v_j)$ for some $1 \leq l \leq m$ and $1 \leq k \leq p$ where $(v_i, v_j) \in E$. But ϕ' assigns 3 colors to each such path and hence every cycle uses at least 3 colors. Moreover, the transformation from G to G' can obviously be done in P-time.

Finally we show that if G' can be assigned a cyclic p-coloring then G can be p-colored. Assume that ϕ' is a cyclic coloring of G'. Define

$$\phi\colon \phi(v_i) = \phi'(v_i), \quad 1 \leq i \leq n$$

We claim that ϕ is a p-coloring of G. Suppose instead that $\phi(v_i) = \phi(v_j)$ where $e_l = (v_i, v_j) \in E$. Then ϕ' must assign a different color to each w_k^l, $1 \leq k \leq p$; otherwise, there is a bi-colored cycle in G_l'. But it follows that ϕ' uses at least $p+1$ colors, a contradiction. •

The proof above has actually established a stronger result than indicated by the statement of Theorem 3.1, since the constructed graph G' is bipartite.

Corollary 3.2: The cyclic coloring decision problem on bipartite graphs is NP-complete. •

Since the *cyclic coloring problem* is the optimization version of $CCDP$, it follows that it cannot be an easier problem. Hence the cyclic coloring problem is NP-hard (even if we restrict our attention to bipartite graphs).

There is a marked similarity between the proof given above and the NP-completeness proof provided by Coleman and Moré [1984] with respect to the path coloring problem (symmetric **direct** problem). Indeed it turns out that the transformation given above will also establish that the path coloring decision problem is NP-complete. (Recall: $\phi\colon V \to \{1,2,...,p\}$ is a *path p-coloring* of a graph G if ϕ is a p-coloring and if ϕ is not a 2-coloring for any path in G of length 3 edges).

We conclude this section with a short discussion on the relationship between path colorings and cyclic colorings. Let $\chi(G)$, $\chi_\pi(G)$, $\chi_o(G)$ denote the chromatic number, the path chromatic number, and the cyclic chromatic number of graph G, respectively. That is, $\chi(G)$ is the smallest integer p such that G has a p-coloring. Similarily, $\chi_\pi(G)$ [$\chi_o(G)$] is the smallest integer p such that G has a path p-coloring [cyclic p-coloring].

The first observation is that a path coloring is a cyclic coloring. To see this consider any cycle O in G and suppose that ϕ is a path coloring. Clearly if O has only three edges then, since ϕ is a coloring, O must be assigned 3 colors. If O has more than 3 edges then O contains a path connecting 4 distinct vertices and hence at least 3 colors are assigned by ϕ. Therefore,

$$\chi_o(G) \le \chi_\pi(G) \tag{3.1}$$

for any graph G.

Of course a cyclic coloring is not necessarily a path coloring: a cycle O of arbitrary large circumference needs only 1 vertex to be assigned a third color and effect a valid cyclic 3-coloring however this assignment is not a valid path coloring in general. This raises an interesting question: how large can $\dfrac{\chi_\pi(G)}{\chi_o(G)}$ be ? This ratio can be arbitrarily close to 2 for band graphs however we have been unable to prove (or disprove) that this is an upper bound.

Finally, since every cyclic coloring of G is a coloring of G, and every coloring of G^2 is a path coloring of G, we can stretch both ends of (3.1) to get

$$\chi(G) \le \chi_o(G) \le \chi_\pi(G) \le \chi(G^2) \tag{3.2}$$

Note that a partition that induces a direct method that ignores symmetry is equivalent to a coloring of G^2 and has at least $\chi(G^2)$ groups (Coleman and Moré [1983]); a partition that induces a direct method that uses symmetry is equivalent to a path coloring of G and has at least $\chi_\pi(G)$ groups; a partition that induces a substitution method is equivalent to a cyclic coloring and has at least $\chi_o(G)$ groups. One final comment on (3.2): each inequality can be made strict by choosing appropriate graphs.

4. Algorithms

The NP-completeness result of the previous section indicates that an efficient heuristic, or approximation scheme, is required. In particular, since it is not crucial that the absolute **fewest** groups be found (though it is desirable), we are willing to settle for an efficient procedure that produces near optimal results in practise. Indeed, such procedures have been suggested by Powell and Toint [1979] and Coleman and Moré [1984]. Furthermore, Coleman and Moré report extensive experimental results. In this section we will interpret such procedures in the light of the new characterization described in this paper. In addition, we will discuss an important computational concern: Given a substitutable partition, is it possible to recover the matrix unknowns (ie. solve for the edges) in an amount of space proportional to the number of matrix unknowns (ie. the number of edges)?

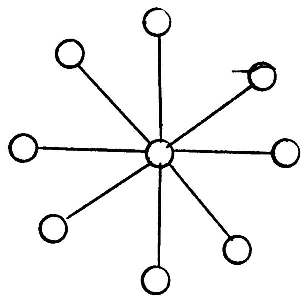
We wish to obtain a cyclic coloring of $G(A)$ using few colors. Since efficient heuristic approaches to the ordinary graph coloring problem (ie. no cyclic restriction) are available, a natural approach is to transform our problem to a general graph coloring problem. In particular, consider adding edges to the given graph $G = (V,E)$ to obtain a completed graph $\overline{G} = (V,\overline{E})$ such that a coloring of $\overline{G}$ is a cyclic coloring of G. Consider the following

```
Algorithm add_edge
  let π:V → {1,...,n} be an invertible map, Ē ← E
  for i = n,...,2 do
    if v_j,v_k are neighbours of π^{-1}(i) in G and π(v_j), π(v_k) < i then
      Ē ← Ē ∪ {(v_j,v_k)}
    fi
  od
end add_edge
```

To see that *add_edge* does the job consider any cycle O in G. Let v_i be the vertex of largest value π on O and let v_j, v_k denote the neighbours of v_k on O. Clearly $(v_j,v_k) \in \overline{E}$ and hence O will need at least 3 colors when $\overline{G}$ is colored.

It is clear that the initial ordering π will affect the resulting graph $\overline{G}$ and consequently the number of colors used. For example, if G is the star graph

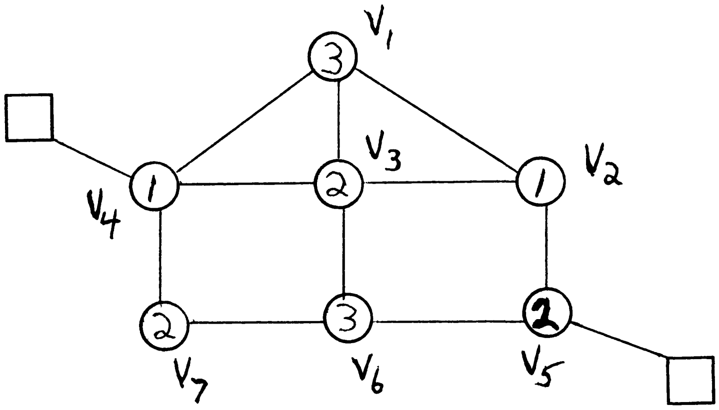
$G=$



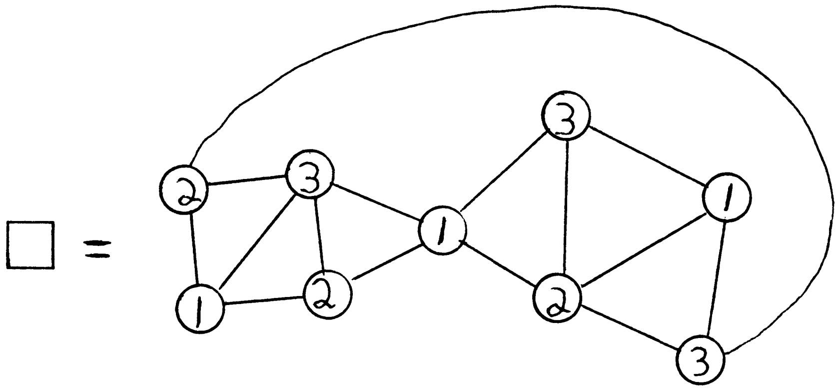
and if the center vertex is ordered last, then $\overline{G}$ is a complete graph. On the other hand, if the center vertex is ordered first, then $\overline{G} = G$. A successful heuristic labelling rule, suggested by Powell and Toint, is the following. Assume that the vertices $\pi^{-1}(n),...,\pi^{-1}(n-k)$ have been found. Choose as the vertex to be ordered $n-k-1$, the vertex of smallest degree in $G - \{\pi^{-1}(n),...,\pi^{-1}(n-k)\}$. This algorithm is known as the smallest last ordering (*slo*) and has a number of interesting properties. For further information consult Coleman and Moré [1984] and Matula and Beck [1983].

The algorithms of Coleman and Moré [1984] and Powell and Toint [1979] both implicitly perform ***add_edge / slo*** followed by a $\overline{G}$-coloring step. Here they differ: the latter authors apply colors in a greedy fashion by considering the nodes in the given order (ie. $\pi^{-1}(1),\ldots,\pi^{-1}(n)$), Coleman and Moré apply a greedy algorithm over several different (cleverly chosen!) orderings. It has been proven that the coloring problem restricted to the class of graphs derived from the ***add_edge / slo*** completion process is NP-complete. However, an important question remains: Does an **optimal** coloring of such a completed graph always solve the cyclic coloring problem? If the answer is yes then one may conclude that it is not necessary to consider algorithms outside this framework. The answer is no.

To see that the cyclic coloring problem may not be solved by an **optimal** coloring of a completion produced by ***add_edge / slo*** consider the following example.



The square vertices represent any cyclic 3-colorable graph where every vertex is of degree at least 3. For example,



The assigned coloring shows that G is cyclic p-colorable. Consider now *add_edge / slo*. Clearly vertex v_7 will be ordered last by *slo* since every other vertex degree is 3 or more. Furthermore v_6 will be labelled second last. It follows that the completed graph will include edges (v_4,v_6) and (v_3,v_5). Therefore v_2,v_3,v_4 must use 3 distinct colors and hence v_1 **must** be assigned a fourth color when $\overline{G}$ is colored.

This example suggest that it may be worthwhile investigating heuristic algorithms for the partition/substitution problem, based on the cyclic coloring characterization perhaps, but not of the *add_edge / slo* variety. At this point we do not know whether there is a practical gain to be made; the answer lies with further experimentation.

One final observation before discussing the solution process: A slight modification of the algorithm *add_edge* yields a procedure for the path coloring problem (symmetric direct method). In particular, change the conditional to read

'if v_j, v_k are neighbours of $\pi^{-1}(i)$ in G and $\pi(v_k) < i$ then '

and it follows that a color assignment of $\overline{G}$ is a path coloring of G. (To our knowledge this heuristic has not been suggested or experimented with previously.) If the conditional is further changed to read

'if v_j, v_k are neighbours of v_i in G then '

it follows that a coloring of $\overline{G}$ is a coloring of G^2 (and hence is also a path coloring of G).

An important computational concern is this: Given that ϕ: $V \rightarrow \{1,...,p\}$ is a cyclic coloring, is it possible to compute the actual matrix elements in space proportional to $|E|$? It turns out that we can answer this question in the affirmative without imposing any additional structure on ϕ. In particular we do not assume that ϕ is necessarily consistent with the algorithm *add_edge / slo* (Coleman, Garbow, and Moré [1984b] discuss, in detail, a FORTRAN 77 implementation of *add_edge / slo*, followed by a graph coloring step. The substitution process operates in space $O(|E|)$ however it relies heavily on the regular matrix structure produced by *add_edge / slo*.)

We will assume that H is stored as a sparse matrix. Rather than define particular data structures, we will assume that

i) indices of the nonzeroes of H as well as the nonzero values H_{ij}, are stored in space $O(|E|)$

ii) it takes time $O($ # nonzeroes in row i of H $)$ to perform each of the following operations

· $H_{ij} \leftarrow x$ (assign the value x to element (i,j)) of H

· list all nonzero column indices in row i

· determine if (i,j) is a nonzero location

Clearly there are a number of possible data structures that would allow these conditions to be met.

The first job is to determine $Hd_j = \nabla f(x+d_j) - \nabla f(x) \triangleq u_j$ and to save the significant information (nonzeroes), for $j=1,...,p$, where p is the number of colors used by ϕ. The vector d_j must be consistent with the color j: if S_j is the set of columns (nodes) in the j^{th} group (color) then $d_j = \sum_{i \in S_j} h_i e_i$, where h_i is the steplength associated with column i. Assume that we have the vector u_j on tap. If $(u_j)_i$ is a nonzero then this quantity is stored as follows:

for each k such that H_{ik} is a nonzero do
  if $\phi(v_k) = j$ then $H_{ik} \leftarrow (u_j)_i$ fi
od

We note that it is not really necessary to replicate the information in H as we have done here; however, not doing so requires a more complicated indexing scheme than we wish to describe here. When the process is complete, H is fully assigned but the numbers do not correspond to the actual matrix quantities: we must now effect a substitution process.

Let us consider the general step. Consider any vertex v which is incident to an unresolved edge, (v,w). Suppose that $\phi(v) = r$ and $\phi(w) = s$. We must now 'grow' the r,s tree rooted at v.

```
Algorithm grow_tree(v,r,s)
  V_0 ← {v}, V_T ← V_0, E_T ← ∅
  i ← 0
  while { V_i ≠ ∅ } do
    V_{i+1} ← ∅
    for each w ∈ V_i do
      if { (w,z) ∈ E−E_T  and  phi(z) = r or s  and ~ (w,z) ∈ E_T } then
        V_{i+1} ← V_{i+1} ∪ {z}
        V_T ← V_T ∪ {z}
        E_T ← E_T ∪ {(w,z)}
      fi
    od
    i ← i+1
  od
  return (T_{r,s}(v) ≜ (V_T,E_T))
end grow_tree
```

The tree $T_{r,s}$ can obviously be stored in space $O(n)$. It is important to realize that it is never necessary to examine more than 1 tree at a time. In particular, the edges of $T_{r,s}(v)$ can be solved and then the tree can be discarded. Algorithm *solve_tree* can be expanded, just slightly, to accomplish this task. In particular, let us detail ' solve (x_i, y_i)' :

```
Procedure solve(x_i,y_i)
  H_ij ← H_ij / h_j,  H_ji ← H_ij
  for each unresolved neighbour z_i of y_i in G with φ(z_i) = φ(x_i)
    H_jk ← H_jk − h_i * H_ji
  fi
end procedure solve(x_i,y_i)
```

In the above procedure we identify index i with vertex x_i, index j with vertex y_i, and index k with vertex z_i.

We conclude Section 4 with a discussion of the time complexity of the procedures to solve for H (i.e. *grow_tree* / *solve_tree*). Note that each (i,j) edge of $G = (V,E)$ belongs to exactly one bi-colored tree: we can charge to each edge the cost of adding it to its tree (*grow_tree*) and the cost of computing its value (*solve_tree*). It is easy to see that the charge is $O(deg_G(v_i) + deg_G(v_j))$, where $deg_G(v)$ is the number of neighbours of vertex v in graph G, and the total time bound is

$$O(\sum_{(i,j) \in E} (deg_G(v_i) + deg_G(v_j)) = O(\sum_{i \in V} (deg_G(v_i))^2)$$

5. Concluding Remarks

We have analyzed a class of methods for estimating sparse Hessian matrices, namely, substitution methods. In particular we have shown that there is an easy and elegant graph theoretic characterization of all substitution procedures based on a partition of columns of the symmetric matrix H. This characterization has allowed for a rich understanding of the combinatorial nature of the problem: we have analyzed the complexity of the partition problem, as well as suggesting efficient procedures to effect the substitution process.

We have restricted our attention, in this paper, to substitution procedures based on a **partition** of columns. Indeed this is more restrictive than need be: Powell and Toint provided a simple example demonstrating that allowing the assignment of 1 column to several groups can reduce the number of required gradient evaluations. In fact, a general 'elimination scheme', where columns can belong to several groups, can be applied to the unsymmetric problem. Newsam and Ramsdell [1983] have explored this option (Coleman [1984] summarizes this idea on page 49). While such methods may occasionally yield a reduction in the number of gradient evaluations, it is not clear that they provide a net benefit, in general, since they require the solution of n square dense (but relatively small) systems of equations to recover the true information.

Two other works should be mentioned. Thapa [1984] has also suggested a direct/partition method for estimating sparse Hessian matrices. Goldfarb and Toint [1984] have proposed specific (optimal) substitution procedures for specific common 'mesh structures': such procedures are, of course, efficient algorithms for obtaining and using optimal cyclic colorings for particular regular structures.

We end with a comment on parallelism. There is a high degree of parallelism in the Hessian estimation problem. Specifically, each estimation Hd_j, $j = 1,...,p$ can be done independently, and thus in parallel. Since this work is sometimes the dominant expense in a numerical problem, exploiting this concurrency may be quite profitable. Note that the number of processors would usually be quite modest, even for large problems, since a cyclic coloring typically uses $<< n$ colors. Moreover, the substitution process also allows for parallel computation: each bi-colored tree can be processed entirely independently of the others.

References.

T.F. Coleman [1984], Large Sparse Numerical Optimization, Springer-Verlag, Volume 165 of Lecture Notes in Computer Sciences.

T.F. Coleman and J.J. Moré [1983], "Estimation of sparse Jacobian matrices and graph coloring problems", SIAM Journal on Numerical Analysis 20 187-209.

T.F. Coleman and J.J. Moré [1984], "Estimation of sparse Hessian matrices and graph coloring problems", Mathematical Programming 28 243-270.

T.F. Coleman, B. Garbow and J.J. Moré [1984a], "Software for estimating sparse Jacobian matrices", Transactions on Mathematical Software, to appear.

T.F. Coleman, B. Garbow and J.J. Moré [1984b], "Software for estimating sparse Hessian matrices", Technical Report ANL-84-xx, Argonne National Laboratory (Argonne, Illinois).

A.R. Curtis, M.J.D. Powell and J.K. Reid [1974], "On the estimation of sparse Jacobian matrices", Journal of the Institute of Mathematics and Its Applications 13 117-119.

M.R. Garey and D.S. Johnson [1979], Computers and intractability (W.H. Freeman, San Francisco, CA).

P.E. Gill, W. Murray, M.A. Saunders and M. Wright [1983], "Computing forward-difference intervals for numerical optimization", SIAM Journal on Scientific and Statistical Computing 4 310-321.

D. Goldfarb, and Ph.L. Toint, [1984], "Optimal estimation of Jacobian and Hessian matrices that arise in finite difference calculations", Mathematics of Computation 43 69-88.

D.W. Matula and L.L. Beck [1981], "Smallest-last ordering and clustering and graph coloring algorithms", Journal of the Association for Computing Machinery 30 417-427.

S.T. McCormick [1983], "Optimal approximation of sparse Hessians and its equivalence to a graph coloring problem", Mathematical Programming 26 153-171.

G.N. Newsam and J.D. Ramsdell [1983]. " Estimation of sparse Jacobian matrices", SIAM Journal on Algebraic and Discrete Methods 4 404-418.

M.J.D. Powell and Ph.L. Toint [1979], "On the estimation of sparse Hessian matrices", SIAM Journal on Numerical Analysis 16 1060-1074.

M.N. Thapa [1984], "Optimization of unconstrained functions with sparse Hessian matrices - Newton-type methods", Mathematical Programming 29 156-186.