COMBINATORIAL PROBLEMS IN AUTOMATIC DIFFERENTIATION

PAUL D. HOVLAND AND UWE NAUMANN

Automatic differentiation (AD) is a family of methods for obtaining the derivatives (sensitivities) of functions computed by a program (see [4] for a detailed discussion). AD couples rule-based differentiation of language built-ins (elementary operators and intrinsic functions) with derivative accumulation according to the chain rule of differential calculus. AD gives rise to a variety of combinatorial problems, including graph coloring, enumeration, and graph transformation problems. We provide a survey of combinatorial problems in AD, with a brief overview of popular heuristics, situations where an optimal algorithm is known, and a list of open problems.

1. Graph Coloring. The computation of sparse Jacobians using graph coloring techniques was first examined by Coleman and Moré in the context of finite difference approximations [1]. Later, they extended this technique to exploit symmetry in Hessian computations. These techniques are equally applicable to automatic differentiation. However, the reverse mode of automatic differentiation introduces the unique capability to computed transposed Jacobian times vector products. Coleman and Verma demonstrated how to exploit this capability with a "bi-coloring." Recently, Gebremedhin et al. presented a unifying framework for all of these coloring problems, posing them as distance-2, distance-$\frac{3}{2}$, and acyclic colorings of graphs arising naturally from the Jacobian or Hessian structure [2].

For certain special cases, an optimal or near-optimal coloring is possible. These special cases include distance-2 colorings of Cartesian grids with and without periodic boundaries [3] and acyclic colorings of planar graphs. The existence of (near-)optimal algorithms for other special cases, such as distance-$\frac{3}{2}$ coloring of planar graphs, is an open question.

2. Jacobian Accumulation. The "accumulation" of derivatives by combining partial derivatives according to the chain rule is the source of another combinatorial problem. The associativity of the chain rule leads to exponentially many ways in which to combine the partial derivatives, each with a different computational cost. Algorithms for automatic differentiation are often expressed in terms of a computational graph. A computational graph is a directed acyclic graph (DAG) whose vertices correspond to elementary operators or intrinsic functions. Figure 2.1 shows the computational graph for the following pseudocode.

```
a = cos(x)
c = sin(y)
d = c*y
b = d*y
e = a*b
f = exp(e)
```

If the edges of the computational graph are assigned weights equal to partial derivatives, then the derivative of a dependent variable v_j with respect to an independent variable v_i is the sum over all paths from v_i to v_j of the product of the edge weights along that path [6]. The optimal Jacobian accumulation problem is reduced to finding an optimal order in which to combine pairs of edge weights [5].

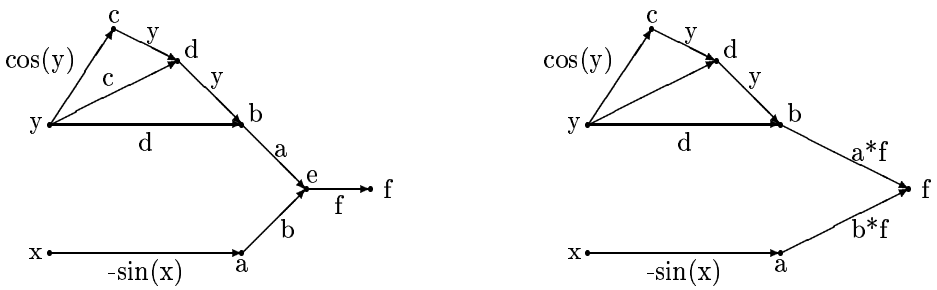
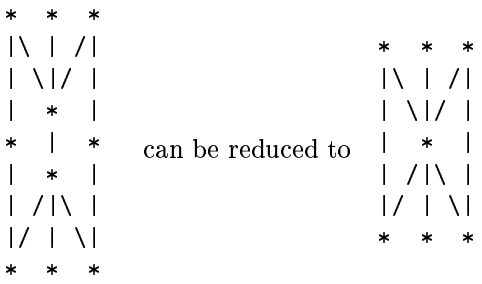


FIG. 2.1. *Computational graph (left) for the simple example, (right) after elimination of vertex e.*

2.1. Vertex elimination. A simplified version of the optimal Jacobian accumulation problem is to find an optimal vertex elimination strategy, where a vertex is eliminated by combining all in edges with all out edges (requiring |in| × |out| multiplications). Figure 2.1 also shows the computational graph after eliminating the vertex `e`.

2.2. Edge Elimination. The computational graph can also be transformed via edge elimination. In this approach, we replace an edge from vertex v_i to vertex v_j with the set of edges from v_i to $\text{succ}(v_j)$ or with the set of edges from $\text{pred}(v_i)$ to v_j, where $\text{succ}(v_j)$ is the set of vertices v_k such that there exists an edge from v_j to v_k and$\text{pred}(v_i)$ is the set of vertices v_k such that there exists an edge from v_k to v_i. There exist graphs for which the optimal edge elimination is superior to the optimal vertex elimination. However, the potential gap between edge and vertex elimination is not yet fully characterized.

2.3. Minimal Representation. A related problem is determining the minimal representation for a Jacobian. Given the vertex or edge elimination transformations described above, it is possible to reduce an arbitrary DAG to a bipartite graph with a number of edges equal to the number of nonzeros in the Jacobian. However, it may also be possible to reduce the graph to a form with few edges. For example, the graph (all edges are implicitly up)



which has fewer edges (8) than the bipartite graph (a complete graph with 9 edges). We present this and related problems, discuss some partial solutions, and describe some open problems.

REFERENCES

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

[2] A. H. Gebremedhin, F. Manne, and A. Pothen, *Graph coloring in optimization revisited*, Technical Report 226, University of Bergen, Dept. of Informatics, Bergen, Norway, Jan. 2002.

[3] D. Goldfarb and P. L. Toint, *Optimal estimation of Jacobian and Hessian matrices that a rise in finite difference calculations*, Mathematics of Computation, 43 (1984), pp. 69–88.

[4] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, SIAM, Philadelphia, 2000.

[5] U. Naumann, *Efficient Calculation of Jacobian Matrices by Optimized Application of the Chain Rule to Computational Graphs*, PhD thesis, Technical University of Dresden, December 1999.

[6] G. Rote, *Path problems in graphs*, in Computational Graphs Theory, Springer-Verlag Computing Supplementum 7, G. Tinhofer, E. Mayr, H. Noltemeier, and M. M. S. in cooperation with R. Albrecht, eds., Springer, 1990.