# IS THE EFFICIENT USE OF THE CHAIN RULE STRAIGHT-FORWARD?

UWE NAUMANN AND JEAN UTKE *

**1. Introduction.** The Jacobian matrix $F'$ of a vector function $F : \mathbb{R}^n \to \mathbb{R}^m$ written as a computer program can be computed with machine accuracy by a source transformation technique known as *automatic differentiation* (AD) [3]. AD is built on the assumption that the arithmetic operators (for example, $*$ and $+$) and intrinsic functions (for example, sin and log) have jointly continuous local partial derivatives at the current argument $\mathbf{x} \in \mathbb{R}^n$. It exploits the chain rule to compute directional derivatives $F' \cdot \dot{\mathbf{x}}$, $\dot{\mathbf{x}} \in \mathbb{R}^n$, and adjoints $(F')^T \cdot \bar{\mathbf{y}}$, $\dot{\mathbf{y}} \in \mathbb{R}^m$, in the *forward* and *reverse* mode, respectively. The forward and reverse modes of AD are two specific ways to exploit the associativity of the chain rule. We discuss the general case, that leads to a variety of combinatorial optimization problems whose (approximate) solution has been shown to have a positive impact on the efficiency of the derivative code.

**2. The Chain Rule in Computational Graphs.** For a given argument $\mathbf{x}$ the computation performed by the program that implements $F(\mathbf{x})$ can be visualized as a directed acyclic computational graph (dag). For example, a function $(v_5, v_6) = F(v_1, v_2)$ that is implemented as $v_3 = \varphi_3(v_1, v_2), v_4 = \varphi_4(v_3), v_5 = \varphi_5(v_4), v_6 = \varphi_6(v_3, v_2)$, where the *elemental* functions $\varphi_j$ represent arithmetic operations or intrinsic functions, leads to the dag shown in FIG. 2.1(a). Under the above assumptions

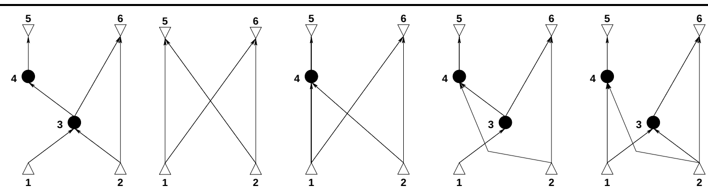


FIG. 2.1. *Transformation of dags*

about the differentiability of the elemental functions the values of the local partial derivatives can be attached to the corresponding edges in the dag. For example, edge $(3, 6)$ is labeled with $c_{6,3} \equiv \frac{\partial \varphi_6}{\partial v_3}$. The Jacobian can be accumulated by transforming the dag into a subgraph of the directed complete bipartite graph $K_{n,m}$. An example is depicted in FIG. 2.1(b). Vertex [4] and edge [6] elimination techniques have been proposed to perform this transformation. For example, FIG. 2.1(c) shows the structural modifications due to the elimination of vertex 3. The *front-elimination* of edge $(2, 3)$ and the the *back-elimination* of edge $(3, 4)$ are illustrated in FIG. 2.1(d) and (e), respectively. The number of "chain rule" operations (fused multiply-adds – fma) is equal to the product of the the number of predecessors and successors for vertex elimination; that is, four in our example. For front- (back-) elimination of edges this cost is equal to the number of successors of the target (predecessors of the source); that is, two in both our examples.

*Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Ave., Argonne, IL 60439, USA, {naumann|utke}@mcs.anl.gov.

The most general elimination technique has been proposed in [7] and is referred to as *face* elimination. This method eliminates edges in the dual computational graph, that is a variant of the directed line graph of the original dag. For example, the dual graph of the dag in FIG. 2.1(a) is depicted in FIG. 2.2(a). Face elimination ultimately leads to the Jacobian in form of a tripartite dag as shown in FIG. 2.2(b) depicting the dual of the dag in FIG. 2.1(b). A single face elimination always performs one fma. Refer to [7] for details on face elimination.

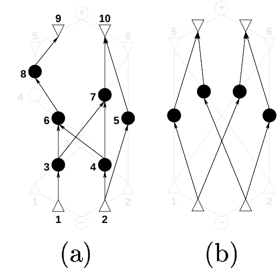

FIG. 2.2. *Dual dag*

**3. Problems and Outline of the Talk.** Vertex, edge, and face elimination are different approaches to the optimal Jacobian accumulation (OJA) problem, that is to minimize the number of fused multiply-add operations required for the accumulation of $F'$. The superiority of edge over vertex elimination in this context has been illustrated in [6]. Moreover, [7] contains an example for the superiority of face over edge elimination. We have not succeeded yet in giving a quantitative characterization of the vertex-edge and edge-face discrepancies.

All three problems are conjectured to be NP-hard. However, no formal proof has been presented so far. Edge and face elimination have been proposed only recently to overcome the discrepancies in optimality that have been outlined above. Before 1999 vertex elimination was considered as the elementary technique for exploitation of the associativity of the chain rule in computational graphs. The apparently closely related problem of minimizing the fill-in under vertex elimination was shown to be NP-complete in [5].

So far, we have been considering the exploitation of the associativity of the chain rule. We are currently investigating the potential impact of the commutativity of scalar multiplication. Our talk shows that the optimal use of the chain rule is not straight-forward at all. We introduce vertex, edge, and face elimination techniques together with their associated combinatorial optimization problems. Various approaches to the approximate solution of these problems are discussed including polynomial deterministic algorithms for relevant special types of dags. The theory is put into the context of generating efficient tangent-linear and adjoint codes automatically by AD.

REFERENCES


[1] G. Corliss, C. Faure, A. Griewank, L. Hascoet, and U. Naumann, editors. *Automatic Differentiation of Algorithms – From Simulation to Optimization*, New York, 2002. Springer.

[2] G. Corliss and A. Griewank, editors. *Automatic Differentiation: Theory, Implementation, and Application*, Proceedings Series, Philadelphia, 1991. SIAM.

[3] A. Griewank. *Evaluating Derivatives. Principles and Techniques of Algorithmic Differentiation.* Number 19 in Frontiers in Applied Mathematics. SIAM, Philadelphia, 2000.

[4] A. Griewank and S. Reese. On the calculation of Jacobian matrices by the Markovitz rule. In *[2]*, pages 126–135, 1991.

[5] K. Herley. A note on the NP-completeness of optimum Jacobian accumulation by vertex elimination. Presentation at: Theory Institute on Combinatorial Challenges in Computational Differentiation, 1993.

[6] U. Naumann. Elimination techniques for cheap Jacobians. In *[1]*, pages 247–253, 2002.

[7] U. Naumann. Optimal accumulation of Jacobian matrices by elimination methods on the dual computational graph. Preprint ANL-MCS/P943-0402, Argonne National Laboratory, 2002. To appear in Math. Prog.