Algorithmic Differentiation: Vertex Elimination in DAGs

Matthias Bentert[*] Alex Crane[†] Pål Grønås Drange[*] Yosuke Mizutani[†]
Blair D. Sullivan[†]

Abstract
We consider graph-theoretic formulations of two problems with wide application in algorithmic differentiation: accumulating a Jacobian matrix while minimizing multiplications (Minimum Cost), and obtaining a minimum-size matrix-free Jacobian representation (Minimum Representation). Both problems ask for a sequence of vertex eliminations in a directed acyclic graph; though a variety of heuristics used in practice are based on these formulations, relatively little is known about their complexity. We prove that both problems are NP-complete, and give an $O^*(2^n)$ algorithm for Minimum Cost. Further, we prove that this running time is essentially the best possible under the Exponential Time Hypothesis. Our results are facilitated by several illustrative structural insights about sequences of vertex eliminations, which also lead to a novel SAT-encoding for Minimum Cost. We conclude with a discussion of open problems related to parameterized algorithms, approximations, and problem variants allowing edge eliminations. In addition to sharing our results, our intent is to spark symbiotic conversations between the algorithmic differentiation and graph algorithms communities.

1 Motivation and Definitions

Throughout scientific computation, physical phenomena are modeled by multivariate vector functions $F\colon \mathbb{R}^n \to \mathbb{R}^m$ which in turn are implemented as numerical programs. A central objective of algorithmic differentiation [14, 18] is to use such a program to compute the Jacobian of the function F; applications abound [4, 5, 7, 9, 10]. This task can be modeled via a directed acyclic graph (DAG) $G = (S \uplus I, E)$ (the *linearized computational graph* of F [13]). The input and output values of the program implementing F are modeled in G by the set S of *sources* and *sinks*. The set I consists of all other (*internal*) vertices, which model intermediate values in the numerical program. Directed edges indicate data dependencies.

Let $\mathcal{P}_{s,t}$ denote the set of all paths from source s to sink t. If we associate to each edge uv the local partial derivative $\frac{\partial v}{\partial u}$, then (as shown by Baur [2]) the chain rule allows us to compute the derivative of t with respect to s:

$$\text{(1)} \qquad \frac{\mathrm{d}t}{\mathrm{d}s} = \sum_{P \in \mathcal{P}_{s,t}} \prod_{uv \in P} \frac{\partial v}{\partial u}.$$

An *elimination* of an internal vertex v is the deletion of v and the creation (if not already present) of directed edges from each in-neighbor to every out-neighbor of v. We write $G_{/v}$ for the resulting DAG. An *elimination sequence* $\sigma = (v_1, v_2, \ldots, v_\ell)$ of length ℓ is a tuple of internal vertices, and we denote by $G_\sigma = (((G_{/v_1})_{/v_2})\ldots)_{/v_\ell}$ the result of eliminating these vertices in the order given by σ. We call σ a *total* elimination sequence if $\ell = |I|$. If σ is a total elimination sequence, then (with appropriate local partial derivative computations or updates during the sequence) G_σ can be thought of as a bipartite DAG representing the Jacobian matrix of the associated numerical program. To reflect the number of multiplications needed to maintain correctness of Equation (1), we say that the *cost* of eliminating an internal vertex v is the product $\deg^-(v) \cdot \deg^+(v)$ (i.e., the *Markowitz degree* of v), and the cost of an elimination sequence is the sum of the costs of the involved eliminations. We can now phrase the problem of computing a Jacobian with few multiplications in purely graph-theoretic terms:

Minimum Cost
Input: A DAG G, and $k \in \mathbb{N}$.
Problem: Does there exist a total elimination sequence with cost at most k?

Closely related is the task of computing a *small* linearized computational graph. Initially introduced in the context of *scarcity* [13], i.e., a setting in which the the structure of the graph encodes information which may be lost in a matrix-representation of the Jacobian, the problem also arises when modeling matrix multiplication using the DAG G as a substitute for the Jacobian; in this setting, the number of required multiplications grows with the number of edges in G [16].

Minimum Representation
Input: A DAG G, and $k \in \mathbb{N}$.
Problem: Does there exist a (not necessarily total) elimination sequence σ such that G_σ has at most k edges?

[*]University of Bergen, Bergen, Norway.
[†]University of Utah, Salt Lake City, Utah.

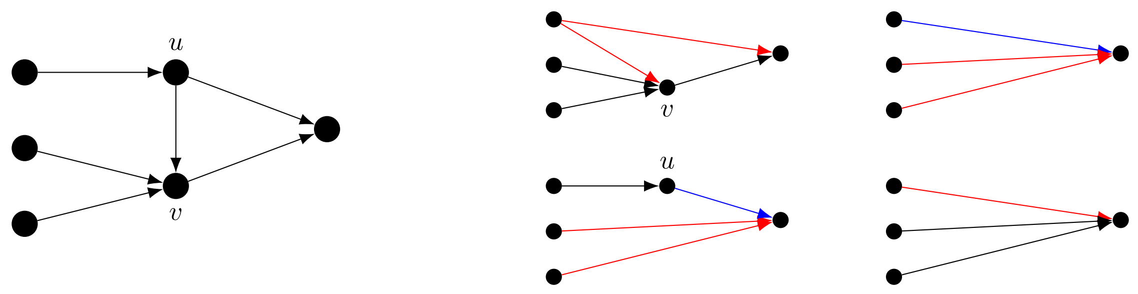

Figure 1: A DAG with two internal vertices (left), and two total elimination sequences (right). The top sequence (u, v) greedily eliminates vertices in ascending order of Markowitz degree, or equivalently (for this example) in topological order. The bottom sequence (v, u) is optimal. By Equation (1), the cost of an elimination is the number of colored edges, which indicate multiplications for newly computed (red) or updated (blue) partial derivatives.

2 Minimum Cost

NP-completeness is known for MINIMUM COST only under certain assumptions about the relationships between local partial derivatives [17]; the complexity of the purely graph-theoretic formulation presented here has remained open since at least 2008 [1, 17]. Despite this, algorithms used in practice are often based on this formulation [8, 11, 12, 19]. These include simple (but not optimal; see Figure 1) heuristics such as eliminating vertices in increasing order of Markowitz degree or in topological order. We contribute a dynamic programming algorithm in the style of Bellmen, Held, and Karp [3, 15]. Intuitively, the insight is that to compute the Markowitz degree of a vertex v at the time of its elimination in a hypothetical total elimination sequence σ, we need only know the *set* U of vertices which precede v in σ; in particular, we do not need to know the order imposed by σ on U. Formally, for an internal vertex v and a set $U \subseteq I$ with $v \notin U$, we define in-reach(G, U, v) as the number of vertices in $V \setminus U$ from which there exists a path to v such that every internal vertex of the path is in U. Similarly, out-reach(G, U, v) is number of vertices in $V \setminus U$ which v can reach using only vertices of U. Thus, if σ is a total elimination sequence and $U \subseteq I$ is the set of vertices preceding v in σ, then the cost of eliminating v in this sequence is the product in-reach$(G, U, v) \cdot$ out-reach(G, U, v). These definitions lead to a dynamic programming algorithm running in $O^*(2^n)$ time, which can be optimized to use only polynomial space at the expense of increased running time [6].

THEOREM 2.1. MINIMUM COST *can be solved in* $O^*(2^n)$ *time and space. Moreover, it can be solved in* $O^*(4^n)$ *time and polynomial space.*

The notion of *reachability* indicated above is also crucial in a novel SAT-encoding. We complement our algorithms by closing the longstanding question of hardness for MINIMUM COST by reducing from VERTEX COVER.

THEOREM 2.2. MINIMUM COST *is NP-complete. Moreover, any algorithm which solves* MINIMUM COST *in* $2^{o(n)}$ *time refutes the Exponential Time Hypothesis.*

3 Minimum Representation

We also resolve the complexity of MINIMUM REPRESENTATION, which has remained open since its introduction in 2005 [13, 16]. Toward this end, our main structural insight is that the order of eliminations does not matter.

LEMMA 3.1. *Let* G *be a DAG and* v_1, v_2 *be two internal vertices of* G. *Then* $(G_{/v_1})_{/v_2} \simeq (G_{/v_2})_{/v_1}$.

Lemma 3.1 says that to solve MINIMUM REPRESENTATION we need only select a *set* $X \subseteq I$ of internal vertices to eliminate, rather than an *order* in which to eliminate them. This distinction is crucial in designing and analyzing our reduction from INDEPENDENT SET.

THEOREM 3.1. MINIMUM REPRESENTATION *is NP-complete.*

4 Open Directions

Despite our progress, many interesting questions remain open. On the theoretical side, perhaps the most immediate is to understand the complexities of of MINIMUM COST and MINIMUM REPRESENTATION under the more expressive *edge elimination* [1] operation. We are also interested in the complexities of these problems in restricted structural classes, particularly DAGs of bounded maximum degree and/or cutwidth. The development of approximation algorithms (or lower bounds) also remains open. In future work we hope to implement the methods we have developed thus far and measure their performance on real data; we also expect that an analysis of the structural properties of DAGs arising from practice will guide further theoretical efforts.

References

[1] S. G. Aksoy, R. Bennink, Y. Chen, J. Frías, Y. R. Gel, B. Kay, U. Naumann, C. O. Marrero, A. V. Petyuk, S. Roy, I. Segovia-Dominguez, N. Veldt, and S. J. Young. Seven open problems in applied combinatorics. *Journal of Combinatorics*, 14(4), 4 2023.

[2] W. Baur and V. Strassen. The complexity of partial derivatives. *Theoretical computer science*, 22(3):317–330, 1983.

[3] R. Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

[4] M. Berz, C. Bischof, G. Corliss, and A. Griewank, editors. *Computational differentiation: techniques, applications, and tools*. Proceedings Series. SIAM, 1996.

[5] C. H. Bischof, H. M. Bücker, P. Hovland, U. Naumann, and J. Utke, editors. *Advances in automatic differentiation*. Lecture Notes in Computational Science and Engineering, Vol. 64, Springer, Berlin, 2008.

[6] H. L. Bodlaender, F. V. Fomin, A. M. Koster, D. Kratsch, and D. M. Thilikos. A note on exact algorithms for vertex ordering problems on graphs. *Theory of Computing Systems*, 50(3):420–432, 2012.

[7] M. Bücker, G. Corliss, P. Hovland, U. Naumann, and B. Norris, editors. *Automatic Differentiation: Applications, Theory, and Tools*. Lecture Notes in Computational Science and Engineering Vol. 50, Springer, Berlin, 2005.

[8] J. Chen, P. Hovland, T. Munson, and J. Utke. An integer programming approach to optimal derivative accumulation. In *Recent Advances in Algorithmic Differentiation*, pages 221–231. Springer, 2012.

[9] G. Corliss, C. Faure, A. Griewank, L. Hascoet, and U. Naumann. *Automatic differentiation of algorithms: from simulation to optimization*. Springer Science & Business Media, 2013.

[10] G. Corliss and A. Griewank, editors. *Automatic differentiation: theory, implementation, and application*. Proceedings Series. SIAM, 1991.

[11] S. A. Forth, M. Tadjouddine, J. D. Pryce, and J. K. Reid. Jacobian code generated by source transformation and vertex elimination can be as efficient as hand-coding. *ACM Transactions on Mathematical Software (TOMS)*, 30(3):266–299, 2004.

[12] A. Griewank and U. Naumann. Accumulating jacobians as chained sparse matrix products. *Mathematical Programming*, 95:555–571, 2003.

[13] A. Griewank and O. Vogel. Analysis and exploitation of jacobian scarcity. In *Modeling, Simulation and Optimization of Complex Processes: Proceedings of the International Conference on High Performance Scientific Computing, March 10–14, 2003, Hanoi, Vietnam*, pages 149–164. Springer, 2005.

[14] A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[15] M. Held and R. M. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962.

[16] V. Mosenkis and U. Naumann. On optimality preserving eliminations for the minimum edge count and optimal jacobian accumulation problems in linearized dags. *Optimization Methods and Software*, 27(2):337–358, 2012.

[17] U. Naumann. Optimal jacobian accumulation is np-complete. *Mathematical Programming*, 112:427–441, 2008.

[18] U. Naumann. *The art of differentiating computer programs: an introduction to algorithmic differentiation*. SIAM, 2011.

[19] J. D. Pryce and E. M. Tadjouddine. Fast automatic differentiation jacobians by compact lu factorization. *SIAM Journal on Scientific Computing*, 30(4):1659–1677, 2008.