An Integer Programming Formulation of the Minimal Jacobian Representation Problem

Paul D. Hovland

May 9, 2016

1 Introduction

Most Jacobian matrices contain hidden structure and can be represented as the sum and/or product of sparse and/or low-rank matrices. Following Griewank [1], the algorithimic differentiation community refers to this sparse and low-rank structure as scarcity or scarsity [2, 3, 4]. Although it can be difficult to recover the scarcity structure from a particular Jacobian matrix, the structure is more apparent in the computational graph used to compute the Jacobian. The computational graph is the directed acyclic graph representing the function being differentiated, augmented with edge weights corresponding to partial derivative values. Then, the value of $J_{ij} = \frac{\partial y_i}{\partial x_j}$ is the sum over all paths from x_j to y_i of the product of the edge weights along the path. See [4] for more details. In this representation, a small number of edges at a given depth represents sparsity and a small number of vertices at a given depth represents low rank structure. Parallel subgraphs are added and sequential subgraphs are multiplied.

2 Vertex elimination on the computational graph

The computationational graph can be transformed into an equivalent graph via one of several different transformations, including vertex elimination, edge elimination, face elimination, edge normalization, edge rerouting [1, 4, 5]. In vertex elimination, a vertex v_k is eliminated by multiplying all incoming edges by all outgoing edges. Then, for each predecessor $v_i \in \mathcal{P}(v_k)$ and successor $v_j \in \mathcal{S}(v_k)$ we increment the edge weight $w_{ij}+ = w_{ik}w_{kj}$, adding a new edge E_{ij} if necessary. Typically, vertex elimination proceeds until all intermediate vertices have been eliminated, yielding a bipartite graph with only input (independent variable) and output (dependent variable) vertices and edges representing nonzero Jacobian entries, with value equal to the corresponding edge weight. However, in order to exploit scarcity, we seek the sequence of vertex eliminations that yields the equivalent computational graph with the smallest possible number of edges. Figure 1 shows an initial graph, the equivalent bipartite graph, and the equivalent graph with the minimal number of edges, obtained by eliminating vertices v_3 and v_4 but not v_5. The initial and bipartite graphs have 9 edges while the minimal representation has 6 edges, corresponding to the outer product of two vectors of length 3, or a 3×3 rank one matrix.

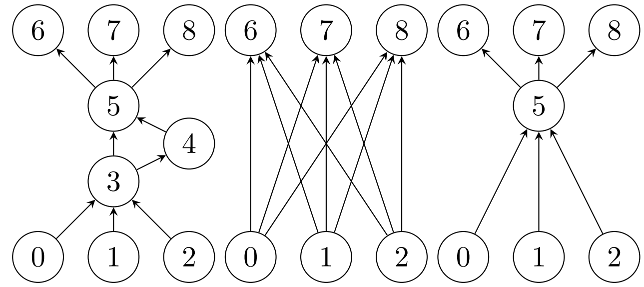

Figure 1: A computational graph, the equivalent bipartite graph, and the equivalent graph with the minimal number of edges.

3 An integer programming formulation of the minimal Jacobian representation problem.

We have developed an integer programming model whose solution provides the vertex elimination sequence that minimizes the number of edges in the resultant computational graph. The model is considerably simpler than previous models developed in order to minimize the number of operations required to completely transform a computational graph to bipartite form [6], since the number of multiplications does not need to be tracked. Key to the model is permitting steps where zero vertices are eliminated. This permits

```
fa(i,j,k)$(given[i,j] and (ord(k) eq 1))..  x[i,j,k] =e= given(i,j);
fb(k)$(ord(k) gt 1)..  sum(i, v[i,k]) =l= 1;
fg(i)..  sum(k$(ord(k) gt 1), v[i,k]) =l= 1;
fc(k)..  sum(i$independent[i], v[i,k]) =e= 0;
fd(k)..  sum(i$dependent[i], v[i,k]) =e= 0;
fe(i,j,k)$(ord(k) gt 1).. x[i,j,k] =g= x[i,j,k-1] - v[i,k] - v[j,k];
ff(i,j,k,l)$(ord(k) gt 1).. x[i,j,k] =g= x[i,l,k-1] + x[l,j,k-1] + v[l,k] - 2;

variable obj; equation objdef; objdef.. obj =e= sum((i,j,k)$(ord(k) gt ninter), x[i,j,k]);
```

Figure 2: GAMS model for the minimal Jacobian representation problem.

incomplete transformations without a need to model the number of transformation steps. The model is as follows:

$$\min \sum_{i,j} X_{ijn} \text{ subject to}$$

$$X_{ij0} = E_{ij} \quad \forall i,j \tag{1}$$

$$\sum_i V_{ik} \leq 1 \quad \forall k \tag{2}$$

$$\sum_k V_{ik} \leq 1 \quad \forall i \tag{3}$$

$$X_{ijk} \geq X_{ij(k-1)} - V_{ik} - V_{jk} \quad \forall i,j,k \tag{4}$$

$$X_{ijk} \geq X_{il(k-1)} + X_{lj(k-1)} + V_{lk} - 2 \ \forall i,j,k,l \tag{5}$$

where X_{ijk} indicates whether edge E_{ij} is present after k elimination steps, E_{ij} is the edge set of the initial graph, V_{ik} indicates whether vertex v_i is eliminated in step k, and n is the number of intermediate vertices. The model enforces the following constraints:

1. initial edge set corresponds to input graph
2. eliminate no more than one vertex at each step
3. do not eliminate a vertex more than once
4. an edge must be preserved unless its source or sink is eliminated
5. an edge must be introduced between the predecessors and successors of eliminated vertices

We have implemented this model in the GAMS modeling language and verified its efficacy on several small test problems. See Figure 2 for the portion of the model representing the objective and constraints.

Acknowledgements

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program, under contract number DE-AC02-06CH11357.

References

[1] Andreas Griewank and Olaf Vogel. Analysis and exploitation of Jacobian scarcity. In Hans Georg Bock, Ekaterina Kostina, Hoang Xuan Phu, and Rolf Rannacher, editors, *Modeling, Simulation and Optimization of Complex Processes*, pages 149–164, Berlin, 2005. Springer.

[2] Andrew Lyons and Jean Utke. On the practical exploitation of scarsity. In Christian H. Bischof, H. Martin Bücker, Paul D. Hovland, Uwe Naumann, and J. Utke, editors, *Advances in Automatic Differentiation*, pages 103–114. Springer, 2008.

[3] Andrew Lyons, Ilya Safro, and Jean Utke. Randomized heuristics for exploiting jacobian scarcity. *Optimization Methods and Software*, 27(2):311–322, 2012.

[4] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* Number 105 in Other Titles in Applied Mathematics. SIAM, Philadelphia, PA, 2nd edition, 2008.

[5] Uwe 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] Jieqiu Chen, Paul Hovland, Todd Munson, and Jean Utke. An integer programming approach to optimal derivative accumulation. In Shaun Forth, Paul Hovland, Eric Phipps, Jean Utke, and Andrea Walther, editors, *Recent Advances in Algorithmic Differentiation*, volume 87 of *Lecture Notes in Computational Science and Engineering*, pages 221–231. Springer, Berlin, 2012.