Exploiting Symmetry for Hessian Computation ⋆

S. Bhowmick[1,2] and P. Hovland[2]

[1] Department of Applied Physics and Applied Mathematics, Columbia University,
200 S.W. Mudd Building, 500 W. 120th Street, New York, NY 10027
[2] Mathematics and Computer Science Division, Argonne National Laboratory,
9700 South Cass Avenue, Argonne, IL 60439-4844.
E-mail {bhowmick, hovland}@mcs.anl.gov

1 Introduction

Hessian computation in automatic differentiation can be implemented by applying elimination operations on a symmetric computational graph. We can exploit the symmetry by identifying pairs of symmetric operations and performing only one operation from each pair. Symmetry exploitation can potentially halve the number of operations. In this presentation, we describe an elimination algorithm that exploits the symmetry of computational graphs.

2 Symmetric Elimination

An important substep in symmetry exploitation is the detection of the axis of symmetry in the computational graph. It may be claimed that since the graph is derived from the function and its gradient, it is possible to detect the axis of symmetry during construction. However, the function variables and their corresponding gradient variables may not be explicitly known, such as when only the computational graph is available. In such cases it is more efficient to determine symmetric pairs based on the graph structure.

We define a computational graph, $G = (V, E)$, to be symmetric if there exists a bijective function σ of V and E, such that ,$\sigma(E) = E$, $\sigma(V) = V$ and for each edge $e \in E$, if the head is $v \in V$ and the tail is $u \in V$, then $\sigma(u)$ is the head and $\sigma(v)$ is the tail of $\sigma(e)$. We define vertices $(v, \sigma(v))$, as a *symmetric vertex pair* and edges $(e, \sigma(e))$ as a *symmetric edge pair*.

Testing symmetry of a general graph is a NP-complete problem [1, 4]. However, computational graphs provide more information about the structure, such as the direction and weight of the edges. Furthermore we are interested only in an axis of symmetry perpendicular to the direction of the edges. Taking this extra information into consideration, we have developed a polynomial time, complexity $O(V^2)$, symmetry detection algorithm for computational graphs. The algorithm has two steps;

Step 1: Group vertices such that two vertices u and v are in the same group if either of the two conditions are true

- *indegree(v)=indegree(u)* AND *outdegree(v)=outdegree(u)*
- *outdegree(v)=indegree(u)* AND *indegree(v)=outdegree(u)*

It is easy to see that if vertices a and b are symmetric pairs, then they must be in the same group.
Step 2: Subdivide each group further, based on edge weights and neighbors of the vertices until there are only two vertices left in each group.

It can be proved that this algorithm converges in $V/2$ steps, and the vertices in each group form a symmetric pair. Once symmetry is determined, symmetric edge elimination is implemented as follows; for each edge to be eliminated, identify its symmetric pair and eliminate it. Set the weight of the edges obtained by this elimination equal to the weight of their symmetric counterparts.

3 Results

We have implemented the symmetry exploitation algorithm within the OpenAD [3] framework and tested the algorithm for edge elimination using the Markovitz heuristic [2]. Table 1 lists the graphs used in our test set.

⋆ This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational Technology Research, U.S. Department of Energy under Contract W-31-109-Eng-38.

Name	Vertices	Edges	Independent Variables	Dependent Variables
ex1	10	13	2	1
ag1	12	16	2	1
ag2	22	34	4	2
opt1	26	51	6	1
ex2	40	75	4	2

Table 1. List of graphs used in test set. Columns independent and dependent variables list the number of independent and dependent variables in the function whose Hessian is being calculated. Graphs ex1 and ex2 were created by us, ag1 and ag2 are from [2] and opt1 is formed from an optimization code [5].

A comparison of the number of multiplications using the heuristic with and without symmetry is given in Figure 1.The results show that exploiting symmetry can indeed lower the number of multiplications. The time taken to find an elimination sequence using symmetry exploitation, which includes time for symmetry detection as well as time for finding the sequence, is comparable to the time taken when only the heuristic is used.

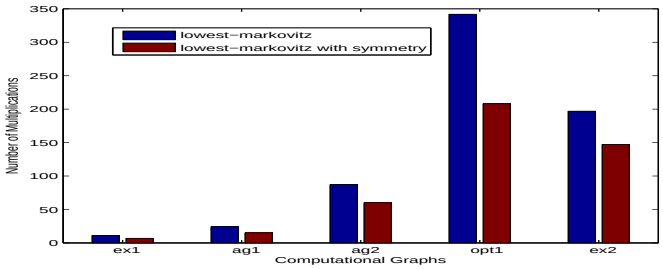

Fig. 1. Comparison of the number of multiplications required for calculating the Hessian in computational graphs of different sizes.

4 Conclusions

The results show that symmetric elimination is successful in lowering the number of multiplications. However, the values are much higher than the theoretical 50% bound, which indicate that it might be instructive to investigate more sophisticated symmetric elimination techniques.We also plan on investigating the existence of a tighter bound on the number of reductions based on the number and degree of edges that cross the axis of symmetry. Other future research plans include implementing symmetric strategies for vertex and face eliminations.

References

1. Hubert De Fraysseix. An heuristic for graph symmetry detection. *Lecture Notes in Computer Science, Proceedings of the 7th International Symposium on Graph Drawing*, 1731:276–285, 1999.
2. A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, 2000.
3. J.Utke. OpenAD: Algorithm implementation user guide. Technical Report ANL/MCS–TM–274, Argonne National Laboratory, IL, 2004.
4. J. Manning. Geometric symmetry in graphs, 1990. Ph.D. Thesis, Purdue University, New York.
5. T. S. Munson. Mesh shape-quality optimization using the inverse mean-ratio metric. Preprint ANL/MCS-P1136-0304, Argonne National Laboratory, Argonne, Illinois, 2004.