Reduction of Edge Labeled DAGs through Constant Folding

Uwe Naumann, Jean Utke

We present intermediate results for a discrete problem arising in the automatic generation of derivative code. Consider a directed acyclic graph (dag) \( G = (V, E) \) with vertex set \( V = X \cup Z \cup Y \) where \( X \) is the set of minimal vertices, \( Z \) is the set of intermediate vertices, \( Y \) is the set of maximal vertices, and \( E \) is the set of edges \((i, j)\) with \( i, j \in V \). \( X, Y, \) and \( Z \) are pairwise disjoint. The vertices are numbered such that they induce a dependency order; \((i, j) \in E \Rightarrow i < j \). Each edge \((i, j)\) is annotated with a label \( c_{ij} \in \mathbb{R} \). The \( c_{ij} \) can be categorized as either variable or constant. \( G \) is transformed into a bipartite graph by using a sequence \( \sigma \) of edge (front and back) and vertex eliminations.

Figure 1 illustrates the elimination techniques and the corresponding operations on the \( c_{ij} \) for a dag with \( X = \{1, 2\}, \ Z = \{3\}, \ Y = \{4, 5\} \). This transformation of \( G \) into bipartite form represents the accumulation of a Jacobian in the context of automatic differentiation [1] where \( G \) is a computational graph and the \( c_{ij} \) are local partial derivatives. Details on the elimination techniques can be found in [4]. The cost of an elimination sequence \( \sigma \) can be measured as the number \( M \) of scalar multiplications performed. The objective is to find a \( \sigma \) that minimizes the cost, that is, \( M = M(G) \). There is no known polynomial algorithm that solves this problem in general, necessitating the use of often costly heuristics [2].

Let \( S_i = \{ j \in V : (i, j) \in E \} \) and \( P_j = \{ i \in V : (i, j) \in E \} \). A path \( p_{ij} \) is a sequence of vertices \( (i = k_1, \ldots, k_l = j) \) such that \((k_{\nu}, k_{\nu+1}) \in E \) for \( \nu = 1, \ldots, l - 1 \). We define an \( X \)-\( j \)-separating vertex set \( X_j \) such that all paths \( p_{ij} \) for any \( i \in X \) contain a \( k \in X_j \). The following algorithm transforms single expression use (sen) dag \( G \) defined by \( |S_i| = 1 : \forall i \in Z \)

---

1This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38 and by the National Science Foundation's Information Technology Research Program under Contract OCE-0205590, "Adjoint Compiler Technology & Standards" (ACTS).

2Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439-4844, USA; {naumann|utke}@mesa.anl.gov
into bipartite form.

**Algorithm 1 (elimination)** For a given \( \hat{G} \) do the following.
1. \( \forall j \in Z \) in dependency order
   1. Compute a minimal \( X_j \)-separating vertex set \( X_j \).
   2. If \( |X_j| < |P_j| \), then generate the bipartite subgraph
      \[ \hat{G}_j = (\{j\} \cup X_j, \{(i, j) : i \in X_j\}) \]
      by vertex elimination in reverse dependency order.
2. Eliminate the remaining vertices \( \in Z \) in reverse dependency order.

The effect of Algorithm 1 is that all intermediate \( j \in Z \) are eliminated at their lowest possible cost \( |X_j| \). It is a generalization of the respective algorithm introduced in [5]. Without formal proof we propose that Algorithm 1 yields \( M(\hat{G}) \) for seu-dags.

Until now we have not made a distinction between operations involving variable vs. constant labels \( c_{ji} \). For the constant \( c_{ji} \) we distinguish trivial edges \( (c_{ji} \equiv \pm 1) \) and the remaining invariant edges \( (c_{ji} \equiv \text{const}) \). Those can yield trivial multiplications by \( \pm 1 \) or multiplications of two constants.

We refine the objective by allowing to discount such trivial multiplications. The complexity of any known algorithm for approximating optimal elimination sequences depends on \( |E| \). Thus the effort can be lowered by reducing the size of \( G \). We refer to this size reduction as constant folding. It needs to preserve optimality, that is, for the reduced graph \( G_r \) we must get \( M(G_r) \leq M(G) \). For seu-dags we propose the following algorithm.

**Algorithm 2 (seu constant folding)** For a given \( \hat{G} \) do the following.
1. Back eliminate all trivial edges.
2. Back eliminate \((j, k)\) if \((j, k)\) and \((i, j)\) are invariant \( \forall i \in P_j \).
3. Front eliminate all trivial edges \((i, j)\) if \( |P_j| = 1 \) or \( S_j \subseteq Y \).

Algorithm 2 can be proven to preserve optimality for seu-dags. It is not necessarily optimal for general dags as illustrated in Figure 2. Not counting trivial multiplications by one, the back elimination of the trivial edges \((4, 3)\) and \((5, 3)\) in (b) leads to an additional multiplication that is avoided in (c).

Similar observations can be made for the other steps leading to the following modification for a general dag \( G \) that only imposes the single expression use property locally as a prerequisite for the respective target vertices of the edges to be eliminated.

**Algorithm 3 (constant folding)** For a given \( G \) do the following.
1. Back eliminate all trivial edges \((i, j)\) if \( |S_i| = 1 \) or \( |P_i| = 1 \) and \( P_i \subseteq X \).
(2) Eliminate $j$ if $(i,j)$ and $(j,k)$ are invariant for all $i \in P_j$ and $k \in S_j$ and $|S_j| = 1 \lor |P_j| = 1$.

(3) Front eliminate all trivial edges $(i,j)$ if $|P_j| = 1$ or $|S_j| = 1$ and $S_j \subseteq Y$.

The refinement of Algorithm 3 is the subject of ongoing research.

All algorithms proposed here have their practical application in the ACTS project, see http://www.autodiff.org/ACTS. This collaborative effort between MIT, Rice University, and University of Chicago / Argonne National Laboratory covers algorithmic research in the field of automatic differentiation. One of the objectives is the development of a toolset that implements these algorithms in order to automatically generate an efficient adjoint of the MIT general circulation model [3].

Figure 2: non-seu scenario

The submitted manuscript has been created by the University of Chicago as Operator of Argonne National Laboratory ("Argonne") under Contract No. W-31-109-ENG-38 with the U.S. Department of Energy. The U.S. Government retains for itself, and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.
References


