An $O(|V| * |E|)$ Algorithm for Finding Immediate Multiple-Vertex Dominators.*

Stephen Alstrup† Jens Clausen† Kristian Jørgensen†

Abstract

We present an $O(|V|*|E|)$ algorithm for finding immediate multiple-vertex dominators in a graph with vertices V and edges E.

1 Introduction.

Finding dominators in a graph has been investigated in many papers [6, 7, 8, 9, 10] in connection with global flow analysis and program optimization. Recently Gupta extended the problem to find generalized dominators [4, 5], which can be use for e.g. propagating loop invariant statements out of the loop in cases, where no single vertex dominates the loop exit, but where a union of vertices together dominates the exit. Fundamental in connection with generalized dominators are the immediate multiple vertex dominators, *imdoms*.

In [5] the immediate multiple vertex dominator set of a given vertex v, $imdom(v)$, is defined . An $O(n * 2^n * |V| + |V|^n)$ algorithm is given for computing $imdom(v)$ for all vertices, where n is the largest cardinality of any of *imdom*. The algorithm is based on the observation that $imdom(v)$ is a subset of the set of immediate predecessors of v. Hence $imdom(v)$ can be obtained by finding these and checking whether the constraints defining an *imdom* are satisfied for each predecessor in turn. The result is a rather complicated algorithm of high complexity.

In this note we use another approach: Based on the constraints defining $imdom(v)$ we derive a precise characterization of those vertices which belong to $imdom(v)$. This characterization immediately gives an $O(|E|)$ algorithm for the single vertex problem leading to an $O(|V| * |E|)$ algorithm for the computation of $imdom(v)$ for all vertices. Our main contribution is hence to discover the characterization of $imdom(v)$ leading to an effective algorithm rather than the derivation of the algorithm from the characterization.

2 Definitions and previous results.

Let $G(V, E, s)$ be a flow graph [1] with start vertex s. The problem is for all vertices (except the start vertex) to find the immediate multiple-vertex dominator (*imdom*) defined by the following three conditions:

1. $imdom(v) \subseteq predecessors(v) = \{w|(w \to v) \in E\}$.
2. Any path from s to v contains a vertex $w \in imdom(v)$.
3. For each vertex $w \in imdom(v)$ a path from s to v exists which contains w and does not contain any other vertex in $imdom(v)$.

*Keywords: Algorithms, Flow analysis.

†E-mail : `(stephen,clausen,morat)@diku.dk`, Department of Computer Science, University of Copenhagen

In [5] an $O(n * 2^n * |V| + |V|^n)$ algorithm is given where n is the largest cardinality of any *imdom*. Hence n is bounded by the largest in-degree in the graph.

Recently Sreedhar and Gao have developed a new representation for flowgraph analysis called DJ-graphs [3]. Using this representation they have given an $(|V| * |E| * |n|)$ algorithm [2].

In the following we present an $O(|V| * |E|)$ algorithm for the same problem.

3 A Characterization of $imdom(v)$.

In order to determine $imdom(v)$ we derive a characterization of $imdom(v)$ which reduces the problem of computing $imdom(v)$ to a reachability problem.

Proposition *The immediate multiple-vertex dominator of a vertex v, $imdom(v)$, consists of the set of predecessors of v, each of which can be reached from s by a path containing no other predecessor of v.*

Proof: Let $v \in V \backslash \{s\}$. If s is a predecessor of v, then by 2) s - being the only possible candidate on the path (s, v) - belongs to $imdom(v)$, and by 3) no other vertex can then belong to $imdom(v)$. Hence in this case $imdom(v) = \{s\}$.

Suppose now that s is not a predecessor of v. If v is not reachable from s then $imdom(v) = \emptyset$ by 3). Otherwise consider any path $P = s, v_1, \cdots v_k, w$ from s to a vertex $w \in predecessors(v)$ for which all $v_i \notin predecessors(v)$. By 1), w is the only possible vertex on P which belongs to $imdom(v)$, so by 2) $w \in imdom(v)$. Oppositely, if $w \in imdom(v)$ then by 1) w is a predecessor of v and by 3) a path Q from s to w without other vertices in $imdom(v)$ exists. Therefore Q cannot contain any other predecessor of v since the first such predecessor would belong to $imdom(v)$ by the previous argument. Hence the set $imdom(v)$ equals the set of predecessors of v, each of which can be reached from s by a path containing no other predecessor of v. $\square$

4 The Algorithm.

To compute $imdom(v)$ we proceed as follows. For each vertex v in the graph we label all the predecessors of v with label *pred*. We then use any graphsearch-method to label with an additional label *visit* all vertices, which can be reached from the start vertex s when avoiding any vertex labeled *pred* (i.e. avoiding x if x is a predecessor of v). Now $imdom(v)$ is the set of vertices labeled both *pred* and *visit*.

Algorithm : Compute *imdom* for every node in $V \backslash \{s\}$ for the graph $G(V, E, s)$.

For every $v \in V \backslash \{s\}$ do begin *compute $imdom(v)$ *
  For every $w \in V$ do begin *unmark the graph*
    pred-label$(w) := False$; *visit*-label$(w) := False$
  end;
  For every $w \in predecessors(v)$ do *pred*-label$(w) := True$;
  $SearchSet := \{s\}$;
  $imdom(v) := \emptyset$;
  Repeat

Choose $x \in SearchSet$;
  $SearchSet := SearchSet \backslash \{x\}$;
  $visit$-label$(x) := True$;
  If $pred$-label$(x) = False$ then *Search on from x *iff* $x \notin predecessors(v)$*
    $SearchSet := SearchSet \bigcup \{y | y \in successors(x) \wedge visit\text{-label}(y) = False\}$
  Else $imdom(v) = imdom(v) \bigcup \{x\}$;
Until $SearchSet = \emptyset$
end;

Proposition *The algorithm described is an $O(|V| * |E|)$ algorithm for the imdom problem.*

Proof. For each vertex in the graph we decide *imdom* by one search in the graph, hence the algorithm has complexity $O(|V| * |E|)$. $\square$

References

[1] A.V. Aho, R. Sethi, and J.D. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley, 1986.

[2] G.R. Gao, Y.-F Lee, and V.C. Sreedhar. DJ-graphs and their application to flowgraph analyses. Technical Report 70, McGill University, School of Computer Science, ACAPS, May 1994.

[3] G.R. Gao and V.C. Sreedhar. A linear time algorithm for placing ϕ-nodes. In *ACM SIGPLAN-SIGACT Symposium on the Principles of Programming Languages*, January 1995.

[4] R. Gupta. Generalized dominators and post-dominator. In *Ann. ACM Symp. on Principles of Programming Languages*, volume 19, pages 246–257, 1992.

[5] R. Gupta. Generalized dominators. *Information processing letters*, 53:193–200, 1995.

[6] D. Harel. A linear time algorithm for finding dominator in flow graphs and related problems. In *Proc. 17th Annual ACM symposium on theory of computing*, pages 185–194, 1985.

[7] M.S. Hetch and J.D. Ullman. A simple algorithm for global data flow analysis of programs. *SIAM J. Comput.*, 4:519–532, 1975.

[8] T. Lengauer and R.E. Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Trans. Programming Languages Systems*, 1:121–141, 1979.

[9] P.W. Purdom and E.F. Moore. Immediate predominators in a directed graph. *Comm. ACM*, 15(8):777–778, 1972.

[10] R.E. Tarjan. Finding dominators in directed graphs. *SIAM J. Comput.*, 3(1):62–89, 1974.