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

Stephen Alstrup ‡ Jens Clausen ‡ Kristian Jørgensen ‡

Abstract

We present an $O(|V| \times |E|)$ 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 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, $indoms$.

In [5] the immediate multiple vertex dominator set of a given vertex $v$, $indom(v)$, is defined. An $O(n \times 2^n \times |V| + |V|^2)$ algorithm is given for computing $indom(v)$ for all vertices, where $n$ is the largest cardinality of any of $indom$. The algorithm is based on the observation that $indom(v)$ is a subset of the set of immediate predecessors of $v$. Hence $indom(v)$ can be obtained by finding these and checking whether the constraints defining an $indom$ 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 $indom(v)$ we derive a precise characterization of those vertices which belong to $indom(v)$. This characterization immediately gives an $O(|E|)$ algorithm for the single vertex problem leading to an $O(|V| \times |E|)$ algorithm for the computation of $indom(v)$ for all vertices. Our main contribution is hence to discover the characterization of $indom(v)$ 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 ($indom$) defined by the following three conditions:

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

*Keywords: Algorithms, Flow analysis.
‡E-mail: (stephen, clausen, morat)@dku.dk, Department of Computer Science, University of Copenhagen
In [5] an $O(n \cdot 2^n \cdot |V| + |V|^n)$ algorithm is given where $n$ is the largest cardinality of any $\text{imd}(v)$. 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 $(|V| \cdot |E| \cdot |n|)$ algorithm [2].

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

3 A Characterization of $\text{imd}(v)$.

In order to determine $\text{imd}(v)$ we derive a characterization of $\text{imd}(v)$ which reduces the problem of computing $\text{imd}(v)$ to a reachability problem.

Proposition The immediate multiple-vertex dominator of a vertex $v$, $\text{imd}(v)$, 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 $v \in V \setminus \{s\}$. If $s$ is a predecessor of $v$, then by 2) $s$ - being the only possible candidate on the path $(s,v)$ - belongs to $\text{imd}(v)$, and by 3) no other vertex can then belong to $\text{imd}(v)$. Hence in this case $\text{imd}(v) = \{s\}$.

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

4 The Algorithm.

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

Algorithm: Compute $\text{imd}$ for every node in $V \setminus \{s\}$ for the graph $G(V,E,s)$.

1. For every $v \in V \setminus \{s\}$ do begin *compute $\text{imd}(v)$ *
2. For every $w \in V$ do begin *unmark the graph*
3. $\text{pred-label}(w) := \text{False}$; $\text{visit-label}(w) := \text{False}$
4. end;
5. For every $w \in \text{predecessors}(v)$ do $\text{pred-label}(w) := \text{True}$;
6. $\text{SearchSet} := \{s\}$;
7. $\text{imd}(v) := \emptyset$;
8. Repeat
9. Choose \( x \in \text{SearchSet} \);
10. \( \text{SearchSet} := \text{SearchSet}\backslash\{x\} \);
11. visit-label\( (x) := \text{True} \);
12. If \( \text{pred-label}(x) = \text{False} \) then *Search on from \( x \) iff \( x \not\in \text{predecessors}(v) \)*
13. \( \text{SearchSet} := \text{SearchSet} \cup \{y | y \in \text{successors}(x) \land \text{visit-label}(y) = \text{False}\} \)
14. Else \( \text{imidom}(v) = \text{imidom}(v) \cup \{x\} \);
15. Until \( \text{SearchSet} = \emptyset \)
16. end;

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

Proof. For each vertex in the graph we decide \( \text{imidom} \) by one search in the graph, hence the algorithm has complexity \( O(|V| \times |E|) \). \( \square \)

**References**


