Upgrading Circuit Modules To Improve Performance+

Doowon Paik Sartaj Sahni

University of Minnesota University of Florida

Abstract

We consider the problem of selectively upgrading some of the modules in a circuit so as to meet a specified performance level. The upgrading of a module involves replacing it with an equivalent module with zero delay (effectively) and this replacement has a cost or weight associated with. We show that some versions of the minimum cost upgrading problem are *NP*-hard while others are polynomially solvable. Several heuristics for the general problem are proposed and evaluated experimentally.

Keywords And Phrases

Module upgrading, performance, delay, complexity, *NP*-hard, heuristics

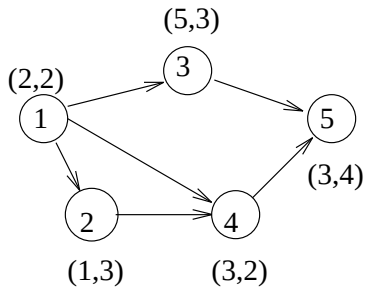
+ This research was supported, in part, by the National Science Foundation under grant MIP86-17374.

1. Introduction

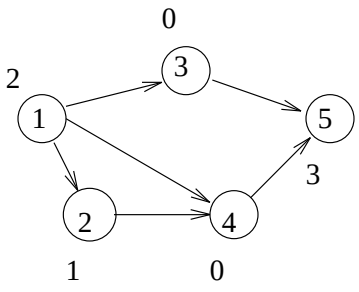
Often, a circuit can be modeled as a directed acyclic graph (dag) G in which there is a delay, $d(v)$, associated with each vertex, v [CHAN90, GHAN87, MCGE90]. The dag vertices represent circuit modules while interconnects are modeled by dag edges. The delay, $d(v)$, associated with vertex v of the dag models both the delay of the modules the vertex represents as well as the interconnect delay. The delay of any path in the dag is the sum of the delays of the vertices on the path. The *circuit delay*, $d(G)$, is the delay of the maximum delay path in the dag. In case the modeled circuit delay is more than the acceptable delay, then one may reduce the delay by replacing some of the modules by modules with lower delay. In this paper we consider the case when these replacement modules have a delay that is very much less than that of the modules they replace. In effect, then, the delay of the new modules is zero. This, for example, is the case when the new modules are implemented in a much faster technology than the old ones. There is a cost, $w(v)$, associated with replacing an old module with a functionally equivalent new one. When an old module is replaced by a new one, we shall say that the corresponding dag vertex v has been upgraded. The cost of the upgrade is $w(v)$. The cost of upgrading a set X of modules/vertices is the sum of the costs of the individual upgrades. Let $G|X$ denote the dag obtained by upgrading the vertices in X. Then the delay of $G|X$ is $d(G|X)$ and the cost of the upgrade is $\sum_{v \in X} w(v)$. Throughout this paper, we assume that $d(v)$ and $w(v)$ are positive integers for all v.

As an example, consider the dag G of Figure 1(a). The number inside a vertex is its label. Outside each vertex is a pair of numbers. The first is its delay and the second is the upgrade cost for that vertex. We see that $d(G) = 10$. When $X = \{3, 4\}$, $G|X$ is as shown in Figure 1(b). In this figure we provide only the delay of each vertex. The upgraded vertices (i.e., those in X) have a delay of 0. The cost of the upgrade is 5.

In this paper we study the problem of obtaining a least cost upgrade set X such that $d(G|X) \leq \delta$ for a specified nonnegative integer δ. This problem is called the directed vertex upgrade problem (DVUP). It is easy to see that DVUP is *NP*-hard. For this we simply show how a polynomial time algorithm for DVUP could be used to solve, in polynomial time, the partition problem which is known to be *NP*-hard [GARE79]. In the partition problem we are given n positive integers $a_1, a_2, ..., a_n$ and wish to know if there is a subset $I \subseteq \{1, 2, ..., n\}$ such that



(a) G with $\delta = 6$

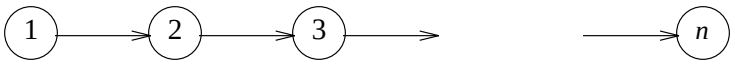


(b) $X = \{3, 4\}$ is the minimal solution

Figure 1: DVUP example (the first(second) coordinate is the delay(weight) of the vertex).

$$\sum_{i \in I} a_i = \sum_{i=1}^{n} a_i / 2$$

Consider the dag below which is just a chain of n vertices. Let $d(i) = w(i) = a_i$, $1 \le i \le n$.



It is easy to see that there is an I such that $\sum_{i \in I} a_i = \sum_{i=1}^{n} a_i / 2$ iff the minimum cost X such that $d(G|X) \le \sum_{i=1}^{n} a_i / 2$ has cost $\sum_{i=1}^{n} a_i / 2$.

The specific results we obtain in this paper are:

1. DVUP is polynomially solvable for dags (circuits) in which all weights and delays are one (Section 2).

2. DVUP is *NP*-hard for dags with all vertex weights one but vertex delays not necessarily one when $\delta \ge 2$. When $\delta = 1$, the problem is polynomially solvable (Section 3).

3. Several heuristics for the general DVUP are proposed (Section 4).

4. An experimental evaluation of the heuristics using ISCAS benchmark circuits [BRGL85, 89] is performed (Section 5).

Related vertex modification problems for dags have been studied in [PAIK90, 91]. In the first of these, vertex splitting was used to model the problem of obtaining an optimal selection of scan flip-flops and in the latter vertex deletion was used to model the problem of optimal placement of signal boosters in lossy circuits.

2. Unit Delay Unit Weight DVUP

A dag $G = (V, E)$ is a *unit delay unit weight dag* iff $d(v) = w(v) = 1$ for every vertex $v \in V$. A subset X of V is *k-colorable* iff we can label the vertices of X using at most k labels and such that no two adjacent vertices have the same label. A *maximum k-coloring* of a dag G is a maximum subset $X \subseteq V$ that is k-colorable. A dag G is *transitive* iff for every $u, v, w \in G$ such that $<u, v> \in E$ and $<v, w> \in E$, the edge $<u, w>$ is also in E. $G^+ = (V, E^+)$ is the transitive closure of (V, E) iff $<u, v> \in E^+$ if there is a path (with at least one edge on it) in G from u to v. Note that if G is a dag then G^+ is a transitive dag.

The unit delay unit weight DVUP for any δ, $\delta \geq 1$ can be solved in $O(n^3 \log n)$ time by using the $O(n^3 \log n)$ maximum k-coloring algorithm of [GAVR87] for transitive dags. Before showing how this can be done, we establish a relationship between the number of vertices of a given set X that can be on any path of G and colorability of X in G^+. Intuitively, X represents the set of vertices that are not upgraded. So, the number of X vertices on a path is the delay for that path.

Theorem 1: Let $G = (V, E)$ be a dag and let $G^+ = (V, E^+)$ be its transitive closure. Let X be a subset of the vertices in V. G has no path with $> \delta$ vertices of X iff X is δ colorable in G^+.

Proof: Suppose that G has no path with $> \delta$ vertices of X. For each $v \in V$ let $m(v)$ be the maximum number of X vertices on any path from a source of G to vertex v. $m(v)$ is given by the recurrence:

$$m(v) = \begin{cases} 0 & v \notin X \text{ and } v \text{ is a source} \\ 1 & v \in X \text{ and } v \text{ is a source} \\ \max\limits_{w \in P(v)} \{m(v)\} & v \notin X \text{ and } v \text{ is not a source} \\ \max\limits_{w \in P(v)} \{m(v)\}+1 & v \in X \text{ and } v \text{ is not a source} \end{cases}$$

where $P(v)$ is the set of immediate predecessors of v and is given by:

$$P(v) = \{w \mid < w,v > \in E\}.$$

Clearly, if G has no path with $> \delta$ vertices of X, it has no vertex v with $m(v) > \delta$. Let $S_i \in X$ be the subset of X that has m value i, $1 \le i \le \delta$. I.e., $S_i = \{v \mid m(v) = i \text{ and } v \in X\}$. We note that $\bigcup_{i=1}^{\delta} S_i = X$ and that if $x \in S_i$, $y \in S_i$, $x \neq y$ then there is no path from x to y (or y to x) in G. To see the latter, observe that if there is a path from x to y (say), then $m(x) < m(y)$ as the path to x with the maximum number of X vertices can be extended to y to get a path with at least one more X vertex (y). As a result of this observation, we conclude that $<x, y> \notin E^+$ and $<y, x> \notin E$ whenever $x \in S_i$, $y \in S_i$, $x \neq y$. So, X is δ colorable in G^+ (vertices in S_i are given the color i, $1 \le i \le k$).

For the reverse, assume that X is δ colorable. Suppose G has at least one path Q that contains $> \delta$ vertices of X. Let the vertices of X on path Q be $v_1, v_2, ..., v_q$, $q > \delta$ and assume that v_i comes before v_{i+1}, $1 \le i \le q$. So, v_i must have a different color from v_j for all i and j such that $i < j$. Hence, $q > \delta$ colors are needed to color $\{v_1, ..., v_q\} \subseteq X$ and X is not δ colorable. This contradicts the assumption on X. Hence if X is δ colorable, G has no path with $> \delta$ vertices of X. □

From the preceding theorem, it follows that if X is a maximum δ-coloring of G^+, $V–X$ is the smallest set such that $d(G|(V–X)) \le \delta$. This implies the correctness of the three step algorithm of Figure 2. The complexity of this is governed by that of step 2 which is $O(n^3 \log n)$. Note that when $\delta = 1$, a maximum δ-coloring is just a maximum independent set and such a set can be found in $O(ne)$ time for transitive closure graphs with n vertices and e edges [GAVR87]. So, the case $\delta \le 1$ can be solved in $O(n^3)$ time as the graph G^+ computed in step 1 of Figure 2 may have $O(n^2)$ edges even though G may not.

step 1: Compute G^+ from G

step 2: Compute X, a maximum δ-coloring of G^+

step 3: $B = V - X$ is the solution for the DVUP instance (G, δ)

Figure 2: Algorithm for unit delay unit weight DVUP.

3. Nonunit Delay Unit Weight DVUP

A dag $G = (V, E)$ is a unit weight dag if $w(v) = 1$ for every $v \in V$. The case when $d(v)$ is also 1 for all v was considered in the previous section. So, in this section we are only concerned with the case of unit weight dags that have at least one vertex with delay > 1. In Section 3.1 we show that the nonunit delay unit weight DVUP can be solved in $O(n^3)$ time when $\delta = 1$ and in Section 3.2, we show that the problem is *NP*-hard when $\delta \geq 2$.

3.1. $\delta = 1$

Let X be a minimum set of vertices such that $d(G|X) \leq 1$. Clearly, every $v \in V$ with $d(v) > 1$ must be in X. For each $v \in V$ with $d(v) > 1$, let $a_1^v, a_2^v, \ldots, a_q^v$ be the vertices such that $< a_i^v, v > \in E$ and let $b_1^v, b_2^v, \ldots, b_r^v$ be such that $< v, b_i^v > \in E$ and let G' be the dag that results when each such v (together with all edges incident to/from v) are deleted from G and all edges of the form $< a_i^v, b_j^v >$ are added. Figure 3 shows the transformation for a single vertex v. To get G', this transformation is applied serially to all v with $d(v) > 1$.

Figure 3: Eliminating a vertex v with $d(v) > 1$.

Let $B = \{v \mid d(v) > 1 \text{ and } v \in V\}$. Let $G' = (V', E')$ and let $C \subseteq V'$ be a minimum vertex set such that $d(G'|C) \leq 1$. It is easy to see that $A = B \cup C$ is a minimum vertex set such that $d(G|X) \leq 1$. C can be obtain in $O(n^3)$ time using the unit delay unit weight algorithm of Section 2 (note that G' is a unit delay unit weight dag), B can be obtained in $O(n)$ time, and G' can be constructed in $O(n^3)$ time. So, the overall complexity of our algorithm to compute X is $O(n^3)$.

3.2. $\delta \geq 2$

To show that non unit delay unit weight DVUP is *NP*-hard for $\delta \geq 2$, we first show the result for $\delta = 2$ and then use this fact to show the result for $\delta > 2$. The proof for the case $\delta = 2$ uses the known [PAIK90] *NP*-hard problem 2-3SAT which is defined below:

Input: A boolean function $F = C_1\ C_2 \cdots\ C_n$ in n variables $x_1, x_2, \ldots, x_n$ such that each clause has either two or three literals. If $|C_i| = 2$ then both literals in C_i are either negated or unnegated. If $|C_i| = 3$ then C_i has at least one negated and one unnegated literal.

Output: "Yes" iff F is satisfiable (i.e., there is an assignment of truth values to $x_1, x_2, \ldots, x_n$ such that all C_i evaluate to true).

To establish the *NP*-hardness of unit weight DVUP with $\delta = 2$, we show how to construct, in polynomial time, an instance $(G_F, 2)$ of this problem and integer k such that there is an X that satisfies:

(a) $d(G_F|X) \leq 2$

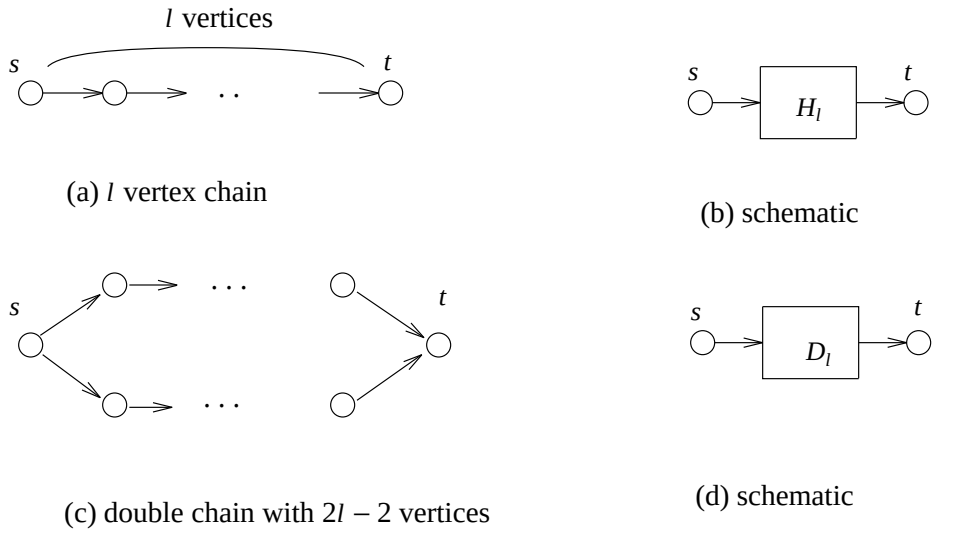
(b) $|X| \leq k$

iff formula F is satisfiable. This construction will make use of variable and clause subassemblies. These are described below.

Variable Subassembly

Let H_l be a chain of l vertices as in Figure 4(a). Each vertex on the chain has unit delay. The schematic for H_l is given in Figure 4(b). The chain source is s and its sink is t. The parallel combination, D_l, of two chains of l unit delay vertices has $2l-2$ vertices and is shown in Figure 4(c). Its schematic is shown in Figure 4(d).

The variable subassembly, $VS(i)$, for variable x_i is D_3 with the source and sink labeled x_i and $\bar{x}_i$ respectively (Figure 5). Note that each vertex in $VS(i)$ has unit delay. x_i and $\bar{x}_i$ are the *external* vertices of $VS(i)$. One easily sees that $d(VS(i)) = 3$. Hence, if $d(VS(i)|X) \leq 2$, then $|X| \geq 1$. Furthermore, if $|X| = 1$, then $X = \{ x_i \}$ or $X = \{ \bar{x}_i \}$.



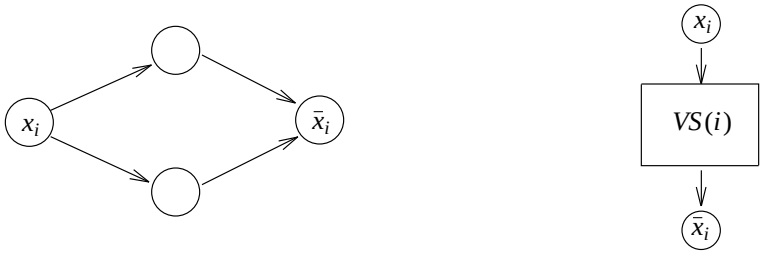
(a) l vertex chain

(b) schematic

(c) double chain with $2l - 2$ vertices

(d) schematic

Figure 4: Single and double chains.



(a) $VS(i)$

(b) Schematic

Figure 5: Variable Subassembly.

Lemma 1: If $d(VS(i)|X) \leq 2$, then $|X| \geq 1$. If $|X| = 1$ and $d(VS(i)|X) \leq 2$, then $X = \{ x_i \}$ or $X = \{ \bar{x}_i \}$ (i.e., X is an external vertex). □

Clause Subassemblies

Our construction of G_F will employ different clause subassemblies for different types of clauses. The subassembly employed will depend on the size of the clause as well as on how many negated literals it contains. The different subassemblies are described below.

Let S_0 be the three vertex dag of Figure 6(a). Vertices a and c have unit delay while the delay of vertex b is 2. The schematic for S_0 is shown in Figure 6(b). We see that $d(S_0) = 3$ and that if $d(S_0|X) \leq 2$, then $|X| \geq 1$. Furthermore, if $|X| = 1$, then $X \subseteq \{b, c\}$. Let l_a (l_b) be the longest delay on any path to a (b) when vertex set $X \subset \{b, c\}$ is upgraded. We see that $(l_a, l_b) = (1, 2)$ when $X = \{c\}$ and $(l_a, l_b) = (2, 1)$ when $X = \{b\}$.

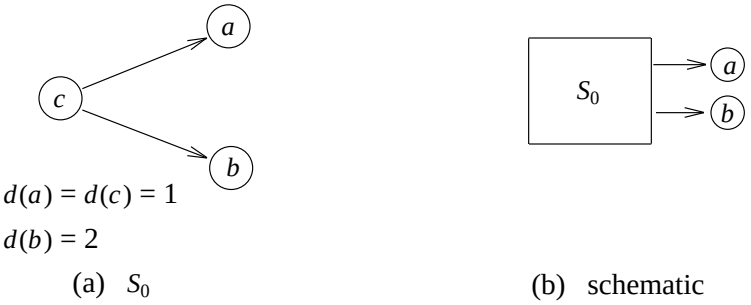
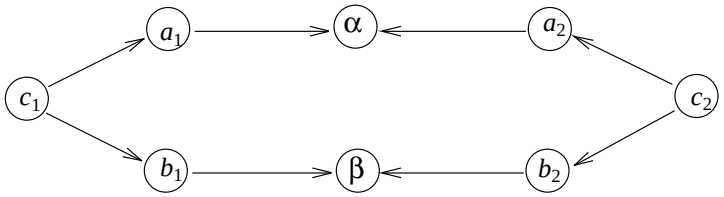


Figure 6: Subgraph S_0.

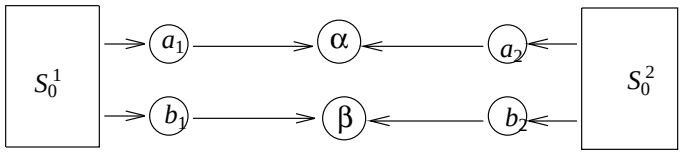
Lemma 2: If $d(S_0|X) \leq 2$, then $|X| \geq 1$. If $|X| = 1$ and $d(S_0|X) \leq 2$, then $X \subset \{b, c\}$ and $(l_a, l_b) = (1, 2)$ when $X = \{c\}$ and $(l_a, l_b) = (2, 1)$ when $X = \{b\}$. □

The subgraph S_1 of Figure 7(a) is obtained by combining two instances of S_0 together via two vertices α and β. The schematic is given in Figure 7(c). The subgraph S_1^R is obtained from S_1 by reversing the direction of all edges. Its delay properties are identical to those of S_1 and its schematic is given in Figure 7(d). α and β are called the *external vertices* of S_1 and S_1^R.

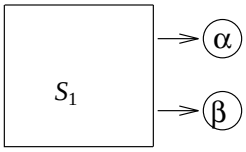
We see that $d(S_1) = 4$. Furthermore, if $d(S_1|X) \leq 2$, then from the construction of S_0 it follows that X must contain at least one of $\{a_1, b_1, c_1\}$ and one of $\{a_2, b_2, c_2\}$. However, this is not sufficient and $|X| \geq 3$. $|X| = 3$ is possible only if $(l_{a_1}, l_{b_1}) = (l_{a_2}, l_{b_2})$. If this is so then β may be added to X as the third vertex in case $(l_{a_1}, l_{b_1}) = (l_{a_2}, l_{b_2}) = (1, 2)$ and α may be added in case $(l_{a_1}, l_{b_1}) = (l_{a_2}, l_{b_2}) = (2, 1)$. However, if $(l_{a_1}, l_{b_1}) \neq (l_{a_2}, l_{b_2})$ then both α and β must be in X to ensure $d(S_1|X) \leq 2$. In this case $|X| > 3$.



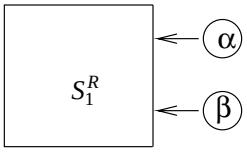
(a) S_1; $d(b_1)=d(b_2)=2$ $d(a_1)=d(a_2)=d(c_1)=d(c_2)=d(\alpha)=d(\beta)=1$



(b) S_1 from two S_0's



(c) Schematic



(d) Schematic for S_1^R

Figure 7: Subgraph S_1.

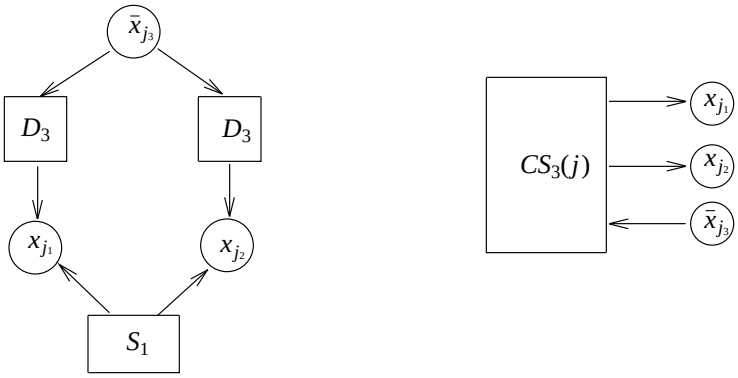
Lemma 3: If $d(S_1|X) \leq 2$, then $|X| \geq 3$. If $|X| = 3$ and $d(S_1|X) \leq 2$, then $X = \{b_1, b_2, \alpha\}$ or $X = \{c_1, c_2, \beta\}$ (i.e., exactly one external vertex is in X). □

The clause subassembly for a two literal clause with no negated literals is simply S_1 and that for a two literal clause with two negated literals is S_1^R. These are, respectively, denoted $CS_2(j)$ and $CS_2^R(j)$ (Figure 8). The schematics are the same as for S_1 and S_1^R except that the α and β vertices have been labeled by the literals of C_j that they represent. The delay properties of $CS_2(j)$ and $CS_2^R(j)$ are the same as those of S_1.

The clause subassembly, $CS_3(j)$, that we use for a three literal clause $C_j = (x_{j_1}+x_{j_2}+\bar{x}_{j_3})$ with exactly one negated literal is given in Figure 9(a). The vertex labeled x_{j_1} is the α vertex of S_1 as well as the sink vertex of D_3; vertex x_{j_2} is both the β vertex of S_1 and the sink of D_3; vertex $\bar{x}_{j_3}$ is the source of both D_3 instances. Note that x_{j_1} and x_{j_2} are sink vertices while $\bar{x}_{j_3}$ is a source. The *external vertices* of $CS_3(j)$ are x_{j_1}, x_{j_2}, and $\bar{x}_{j_3}$.

(a) $CS_2(j)$ (b) $CS_2^R(j)$

Figure 8: Clause subassemblies for two literal clauses.



(a) $CS_3(j)$ (b) schematic

Figure 9: Clause subassembly for a three literal clause with one negated literal.

Lemma 4:

(a) If $d(CS_3(j)|X) \leq 2$, then $|X| \geq 4$.

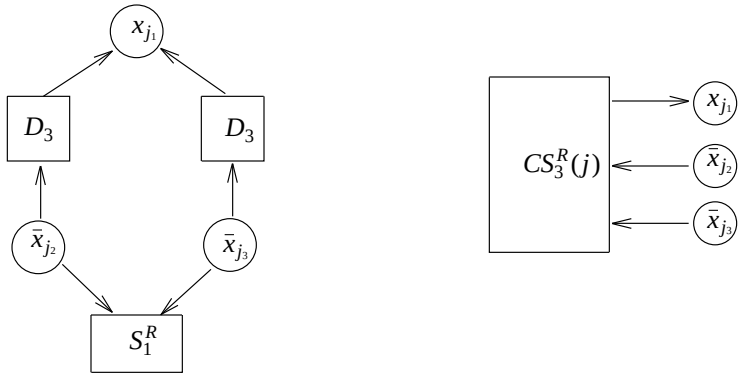
(b) If $|X| = 4$ and $d(CS_3(j)|X) \leq 2$, then $|\ X \cap \{x_{j_1}, x_{j_2}, \bar{x}_{j_3}\}\ | = 2$. I.e., X contains at least two external vertices.

(c) For every $B \subset \{x_{j_1}, x_{j_2}, \bar{x}_{j_3}\}, |B| = 2$, there exists an $X, |X| = 4$ such that $d(CS_3(j)|X) \leq 2$.

Proof: From the construction it follows that if $d(CS_3(j)|X) \leq 2$, then $d(D_3|X) \leq 2$ and $d(S_1|X) \leq 2$. From Lemmas 1 and 3 it follows that X must contain at least one vertex from each of the D_3's and at least three

from S_1. However, from Lemma 3, it also follows that if X contains only three vertices from S_1 then it contains exactly one of x_{j_1} and x_{j_2}. And so X must contain a fourth vertex not in S_1 from one of the D_3's to satisfy Lemma 1. Hence, $|X| \geq 4$. For (b), we note that if all four vertices of X are from S_1, then $\{x_{j_1}, x_{j_2}\} \subset X$ as to get $d(CS_3(j)|X) \leq 2$, X must contain at least one vertex from each of the D_3's. If only three vertices of X are from S_1 then one of x_{j_1} and x_{j_2} must be in X (Lemma 2). If it is x_{j_1} (x_{j_2}) then from Lemma 1 it follows that $\bar{x}_{j_3}$ must be in X so that $d(D_3|X) \leq 2$ for the right (left) D_3 of $CS_3(j)$. (c) can be proved in a similar way. □

When a three literal clause $C_j = (x_{j_1}+\bar{x}_{j_2}+\bar{x}_{j_3})$ has exactly two negated literals, we use the clause subassembly $CS_3^R(j)$ (Figure 10) which is obtained by reversing the direction of all edges in $CS_3(j)$. The schematic S_1^R denotes S_1 with all edges reversed. The delay properties of $CS_3^R(j)$ are identical to those of $CS_3(j)$ and are stated in Lemma 5.



(a) $CS_3^R(j)$ (b) Schematic

Figure 10: Clause subassembly for a three literal clause with two negated literal.

Lemma 5:

(a) If $d(CS_3^R(j)|X) \leq 2$, then $|X| \geq 4$.

(b) If $|X| = 4$ and $d(CS_3^R(j)|X) \leq 2$, then $|\ X \cap \{x_{j_1}, x_{j_2}, \bar{x}_{j_3}\}\ | = 2$. I.e., X contains at least two external vertices.

(c) For every $B \subset \{x_{j_1}, x_{j_2}, \bar{x}_{j_3}\}, |B| = 2$, there exists an $X, |X| = 4$ such that $d(CS_3^R(j)|X) \leq 2$.

Proof: Same as for Lemma 4. □

For any n variable instance $F = C_1\ C_2 \cdots\ C_m$ of 2-3SAT, we construct an instance G_F of unit weight DVUP by using n variable subassemblies (one each variable x_i in F) and m clause subassemblies (one each clause in F). The subassembly used for C_i depends on its type as described above. Vertex x_i ($\bar{x}_i$) of variable subassembly $VS(i)$ is connected to each of the vertices labeled x_i ($\bar{x}_i$) in the clause subassemblies using the double chain D_3. In the case of connecting an x_i of $VS(i)$ to an x_i of a clause, the source of D_3 is the same as the x_i vertex of $VS(i)$ while the sink of D_3 is the same as the x_i vertex of the clause. For an $\bar{x}_i$ connection, the $\bar{x}_i$ vertex of $VS(i)$ is the sink of D_3 and the $\bar{x}_i$ vertex of the clause is the source of D_3. Figure 11 gives G_F for the case $F = (x_1+\bar{x}_2+\bar{x}_4)\ (x_2+x_3+\bar{x}_4)\ (\bar{x}_1+\bar{x}_3)\ (x_2+x_3)$.

Theorem 2: Let F be an n variable m clause instance of 2-3SAT and let G_F be as constructed above. F is satisfiable iff there exists a vertex set X such that $d(G_F|X) \leq 2$ and $|X| = n + 4m - q$, where q is the number of two literal clauses in F.

Proof: From Lemmas 1 – 5 we know that X must contain at least 1 vertex from each $VS(i)$, at least three from each CS_2, and CS_2^R and at least four from each CS_3 and CS_3^R. Since the vertices of the $VS(i)$'s, $CS_2(j)$'s, $CS_2^R(j)$'s, $CS_3(j)$'s, and $CS_3^R(j)$'s are disjoint, $|X| \geq n + 4m - q$. If F is satisfiable then an X satisfying the theorem may be constructed in the following way. Let $x_i = b_i$, $1 \leq i \leq n$, be a set of truth values that satisfy F.

(a) Vertex x_i of $VS(i) \in X$ iff b_i = true.

(b) Vertex $\bar{x}_i$ of $VS(i) \in X$ iff b_i = false.

(c) If C_j is a three (two) literal clause, then from its three (two) external vertices eliminate one that corresponds to a true literal (such a vertex must exist as F is satisfied with the selected truth assignment). The remaining two (one) external vertices plus two more (Lemma 3 and 4) as needed to

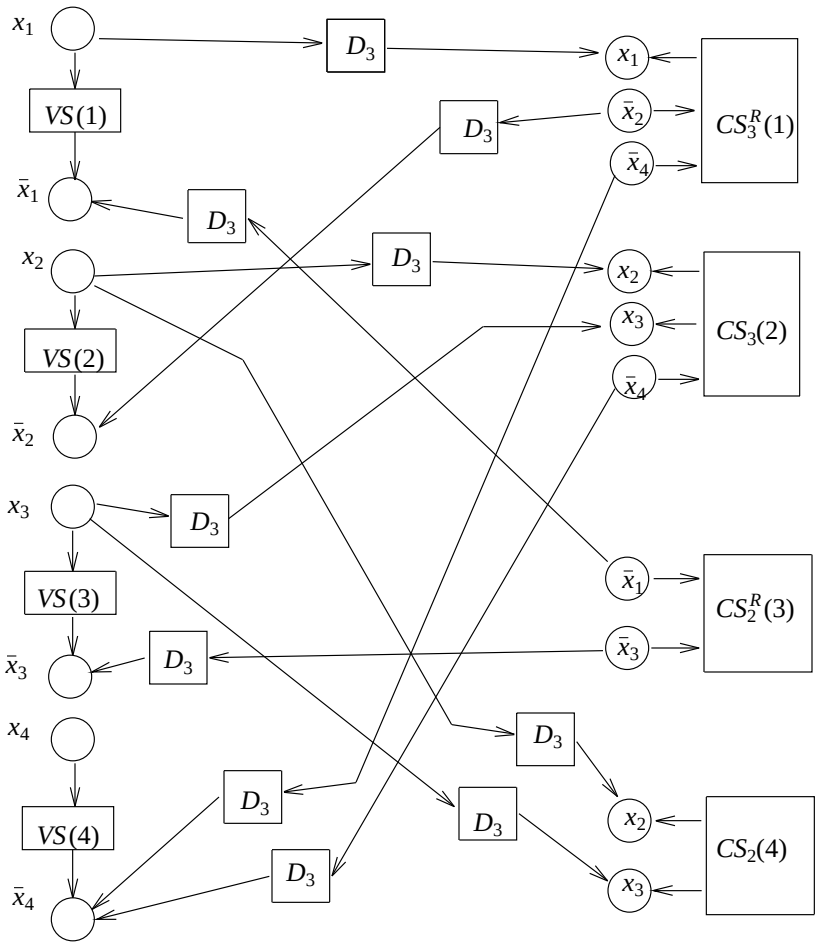


Figure 11: G_F for $F = (x_1+\bar{x}_2+\bar{x}_4)\ (\bar{x}_1+\bar{x}_3+\bar{x}_4)\ (x_1+x_2+x_3)$.

make the delay of the corresponding clause subassembly ≤ 2 are in X.

One may verify that for X constructed as above, $d(G_F|X) = 2$ and $|X| = n + 4m - q$.

Next, suppose that X is such that $|X| = n + 4m - q$ and $d(G_F|X) \leq 2$. For this X must contain the minimum number of vertices needed from each variable and clause subassembly to make the delay of that subassembly ≤ 2. Hence, X contains exactly one vertex from each variable subassembly, exactly three

from each two literal clause subassembly, and exactly four from each three literal clause subassembly. From Lemmas 1 – 5 we see that the vertex from each variable subassembly is an external vertex, exactly one of the vertices in X from each two literal clause subassembly and exactly two from each three literal clause subassembly must be external vertices. If the x_i vertex of $VS(j)$ is in X, set x_i to true otherwise set x_i to false, $1 \leq i \leq n$. Under this truth assignment, each clause of F must be true. To see this, note that from the preceding discussion it follows that one of the external vertices of each clause subassembly is not in X. If this is a sink (source) of a D_3 connector then the corresponding source (sink) must be in X otherwise $d(D_3|X) = 3$. Hence, this literal is true. Note that for each D_3 of G_F one of the source and sink vertices is an external vertex of a variable subassembly and the other is an external vertex of a clause subassembly. □

The *NP*-hardness of non unit delay unit weight DVUP for $\delta = 2$ now follows from Theorem 2, the fact that G_F can be constructed from F in polynomial time, and the fact that 2-3SAT is *NP*-hard.

Theorem 3: Non unit delay unit weight DVUP is *NP*-hard for every δ, $\delta \geq 2$.

Proof: For $\delta = 2$, the theorem has been proved above. For $\delta = q$, $q > 2$ let G be an instance of non unit delay unit weight DVUP with $\delta = 2$. Let G' be obtained from G by attaching to each vertex of G a chain H_{q-2} of $q-2$ vertices (see Figure 12).

(a) A node in a graph (b) After attaching H_{q-2}

Figure 12: Attaching chains to vertices of G.

Let X and X', respectively, be minimum vertex sets such that $d(G|X) \leq 2$ and $d(G'|X') \leq q$. We shall show that $|X| = |X'|$. Since $d(G|X) \leq 2$, $d(G'|X) \leq q$. Hence, $|X'| \leq |X|$. If X' contains a vertex w that is not in G then w must be on some H_{q-2}. Let v be the vertex of G to which the H_{q-2} that contains w was attached in the construction of G'. Let $X'' = X' - \{w\} + \{v\}$. It is easy to see that $d(G'|X'') \leq q$ and $|X''| \leq |X'|$. However, since X' is a minimal set such that $d(G'|X') \leq q$, $|X''| = |X'|$. In this way we can transform X' to X^* such that $d(G'|X^{*)} \leq q$, $|X'| = |X^*|$, and X^* consists solely of vertices of G. So, $d(G|X^{*)} \leq 2$. Hence, $|X'| = |X^*| \geq |X|$. Consequently, $|X'| = |X|$. From this, the observation that G' can be constructed from G in polynomial time, and the fact that unit weight DVUP with $\delta = 2$ is *NP*-hard, it follows that the unit weight DVUP with $\delta = q$ for any $q \geq 2$ is *NP*-hard. □

Note that since the construction used by us generates a multistage graph, DVUP is *NP*-hard even when the dags are restricted to be multistage graphs.

4. Heuristics

We formulate seven heuristics to obtain a low weight set X such that $d(G|X) \leq \delta$. All of these upgrade one vertex at a time and terminate as soon as the delay of the upgraded graph becomes $\leq \delta$. The heuristics differ only in how they select the next vertex to upgrade. The general form of each of the heuristics is given in Figure 13. In the remaining subsections of this section, we describe the criteria used to select the next vertex to upgrade.

```
X := ∅ ; { X is the set of vertices to upgrade }
while d(G|X) > δ do
begin
    Select the next vertex, v , to upgrade;
    X := X ∪ {v};
end;
```

Figure 13: General form of heuristics.

4.1. Heuristic 1 (h1)

For each vertex $v \notin X$ define $N(v)$ to be the number of edges incident to/from v that are on paths with delay > δ. I.e., $N(v) = |\{ <i,v> \in E \mid f(i)+g(v) > \delta \}| + |\{ <v,j> \in E \mid f(v)+g(j) > \delta \}|$ where $f(i)$ is the delay of the maximum delay path in $G|X$ that ends at vertex i and $g(j)$ is the delay of the maximum delay path in $G|X$ that begins at vertex j. The vertex, v, with the largest value of $N(v)/w(v)$ is the next vertex to upgrade.

4.2. Heuristic 2 (h2)

For each vertex $v \notin X$ let $W(v)$ to be the weight of the vertices in $G|(X \cup \{v\})$ that are on paths of delay > δ. I.e., $W(v) = \sum_{y \in S(v)} w(y)$ where $S(v) = \{y \mid y$ is on a path of delay > δ in $G|(X \cup \{v\}), y \notin X, y \neq v\}$. For the next vertex to upgrade, select v such that $W(v)$ is minimum.

4.3. Heuristic 3 (h3)

For each vertex $v \notin X$ let $D(v)$ to be the reduction in $d(G|X)$ that results from changing X to $X \cup \{v\}$. I.e., $D(v) = d(G|X) - d(G|(X \cup \{v\}))$. For the next vertex to upgrade, select v so that $D(v)/w(v)$ is maximum.

4.4. Heuristic 4 (h4)

Let $S = \{s_1, s_2, ..., s_p\}$ and $T = \{t_1, t_2, ..., t_q\}$ be the sets of source and sink vertices of G, respectively. Let $f(u)$ and $g(u)$, respectively, denote the maximum delay of any path in $G|X$ from a source to u and from u to a sink. Define $E(G|X)$ as below:

$$E(G|X) = \sum_{\substack{u \in S \\ g(u) > \delta}} (g(u) - \delta) + \sum_{\substack{u \in T \\ f(u) > \delta}} (f(u) - \delta)$$

E measures (approximately) how far $G|X$ is from having no source to sink path with delay > δ. For each vertex $v \notin X$ we may define $E(G|(X \cup \{v\}))$ in a similar manner. By upgrading vertex v we have come "closer" to the desired state by the amount:

$$\Delta(v) = E(G|X) - E(G|(X \cup \{v\})).$$

The cost of doing this is $w(v)$. For the next vertex v, we pick one that maximizes $\Delta(v)/w(v)$. Note that when G has a single source and a single sink, this selection criterion is the same as that used in heuristic 3.

4.5. Heuristic 5 (h5)

This is similar to heuristic 4 except that the definition of E is changed to include all vertices not in X that are on paths with delay > δ. The new definition is:

$$E(G|X) = \sum_{\substack{u \in V-X \\ g(u)+f(u)-d(u)>\delta}} (g(u)+f(u)-d(u)-\delta)$$

A variation of this heuristic is to do the summation over all vertices $u \in V$ (rather than $u \in V-X$). It was experimentally determined that this does not perform as well.

4.6. Tie Breaker Rule

In case any of the selection criteria results in a tie, the tie is broken by computing min $\{f(u), g(u)\}$ for all the tied vertices (f and g are defined with respect to $G|X$ as in Section 4.1). The tied vertex for which this is maximum is selected (any remaining ties are broken arbitrarily).

4.7. Complexity

For a dag G with n vertices and e edges, $e \geq n$, the time required to select the next vertex using the criterion of heuristics h1 – h7 is, $O(e)$ for h1, and $O(ne)$ for h2 – h5.

5. Experimental Results

The heuristics of the preceding section were programmed in Pascal and evaluated on an Apollo DN3500 workstation. The test circuits used were derived from ISCAS benchmark combinational and sequential circuits [BRGL85, BRGL89]. The sequential circuits were transformed into directed acyclic graphs as in [LEE90]. The characteristics of the two circuits sets are given in Table 1 and 2, respectively. For the sequential circuits, vertex delays in the range 1 – 10 were assigned by us. For the combinational circuits, the vertex delays were obtained by taking the maximum delay on the incomming edges to each vertex. For each circuit, G, we determined the solution obtained by each of the heuristics for $\delta = 0.9d(G)$, $.8d(G)$, $.7d(G)$, $.6d(G)$, $.5d(G)$. The size of the vertex upgrade sets for two of the circuits (s38417, c1908) are given in Tables 3 and 4. These also give the heuristic run time in seconds. Table 5 gives the total number of vertices upgraded for all 23 circuits (for each circuit this is summed over the five δ values used).

As can be seen, heuristics 4 and 5 consistently provide good solutions and their run time is very competitive.

6. Conclusions

We have shown that the min cost upgrading problem is *NP*-hard for multistage dags with unit vertex weights when $\delta \geq 2$. Some other versions of the DVUP problem have been shown to be polynomially solvable. Five heuristics for general dags were proposed and evaluated experimentally. Two of this were determined to be superior to the others.

7. References

[BRGL85] F. Brglez and H. Fujiwara, "A Neutral Netlist of Ten Combinational Benchmark Circuits and a Target Translator in FORTRAN", *Proc. IEEE Symp. on Circuits & Systems*, June 1985 pp. 663-666.

[BRGL89] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits", *Proc. of Intern. Symp. on Circuit & Systems*, May 1989, pp. 1929-1934.

[CHAN90] P. K. Chan, "Algorithms For Library-Specific Sizing Of Combinational Logic", *Proc. 27th DAC Conf.*, 1990 pp. 353-356.

[GARE79] M. R. Garey, and D. S. Johnson, "Computers and Intractability", W. H. Freeman and Company, San Francisco, 1979.

[GAVR87] F. Gavril, "Algorithms For Maximum k-colorings And k-coverings Of Transitive Graphs", *Networks*, Vol. 17, pp. 465-470, 1987.

[GHAN87] S. Ghanta, H. C. Yen, and H. C. Du, "Timing Analysis Algorithms For Large Designs", University of Minnesota, Technical Report, 87-57,1987.

[LEE90] D. H. Lee and S. M. Reddy, "On Determining Scan Flip-flops In Partial-scan Designs", *Proc. of International Conference on Computer Aided Design*, November 1990.

[MCGE90] P. McGeer, R. Brayton, R. Rudell, and A. Sangiovanni-Vincentelli, "Extended Stuck-fault Testability For Combinational Networks", *Proc. of the 6th MIT Conference on Advanced Research in VLSI*, MIT Press, April 1990.

[PAIK90] D. Paik, S. Reddy, and S. Sahni, "Vertex Splitting In Dags And Applications To Partial Scan Designs And Lossy Circuits", University of Florida, Technical Report, 90-34,1990.

[PAIK91] D. Paik, S. Reddy, and S. Sahni, "Deleting Verticies To Bound Path Lengths", University of Florida, Technical Report, 91-4, 1990.

circuit	# vertices	# edges	$d(G)$
s400	173	282	116
s420	37	130	89
s526	27	98	79
s526n	27	98	79
s838	69	266	216
s1423	74	917	160
s5378	233	1314	132
s9234	216	1633	184
s13207	762	3083	214
s15850	608	8562	348
s35932	1777	3380	214
s38417	1396	8754	192
s38584	1448	9471	723

Table 1: Circuit characteristics of modified sequential circuits

circuit	# vertices	# edges	$d(G)$
c432	250	426	57.40
c499	555	928	53.30
c880	443	729	53.00
c1355	587	1064	49.90
c1908	913	1498	76.59
c2670	1426	2076	86.87
c3540	1719	2939	98.69
c5315	2485	4386	99.30
c6288	2448	4800	319.88
c7552	3719	6144	85.30

Table 2: Circuit characteristics (with max of falling and rising delay) of ISCAS combinational circuits.

	# nodes split					run time (sec)				
$\lfloor\delta\rfloor$	h1	h2	h3	h4	h5	h1	h2	h3	h4	h5
$.9d(G)$	24	3	3	3	3	4	54	54	55	55
$.8d(G)$	111	7	10	7	7	18	200	244	232	205
$.7d(G)$	174	16	23	15	14	30	580	811	715	619
$.6d(G)$	218	35	42	27	29	41	1977	2641	1684	1591
$.5d(G)$	268	62	104	49	49	51	3822	7958	4604	7570

Table 3: Results for s38417

	# nodes split					run time (sec)				
$\lfloor\delta\rfloor$	h1	h2	h3	h4	h5	h1	h2	h3	h4	h5
$.9d(G)$	12	3	17	3	3	< 0	20	114	21	24
$.8d(G)$	24	19	56	19	15	1	166	632	186	154
$.7d(G)$	48	67	83	42	39	2	803	1232	688	609
$.6d(G)$	104	123	124	90	76	5	1927	1947	1656	1367
$.5d(G)$	160	180	148	127	103	9	3159	2984	2480	2114

Table 4: Results for c1908

circuit	h1	h2	h3	h4	h5
s400	46	33	57	25	26
s420	34	23	16	16	16
s526	27	16	16	16	16
s526n	27	16	16	16	16
s838	65	62	38	38	38
s1423	46	32	31	27	27
s5378	172	26	51	24	24
s9234	81	37	73	34	34
s13207	128	106	170	81	81
s15850	256	143	217	104	106
s35932	636	527	658	393	393
s38417	795	123	182	101	100
s38584	613	373	380	231	229
c432	28	19	204	22	42
c499	138	257	415	274	173
c880	118	126	268	76	86
c1355	162	303	431	306	204
c1908	348	392	428	281	236
c2670	300	126	1343	71	104
c3540	859	710	1299	356	390
c5315	521	328	995	212	223
c6288	1640	1189	3198	644	1029
c7552	828	564	2577	397	467

Table 5: Total number of nodes upgraded