Optimum Clustering for Delay Minimization*

R. Rajaraman and D. F. Wong
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188

Abstract

This paper addresses the problem of circuit clustering for delay minimization, subject to area capacity constraints. We use the general delay model, for which only heuristic solutions were known. We present an optimum polynomial-time algorithm for combinational circuits under this model. Our algorithm can be generalized to solve the problem under any monotone clustering constraint.

1 Introduction

Circuit partitioning consists of dividing the circuit into parts each of which can be implemented as a separate component (e.g., a chip), that satisfies the design constraints. One such restriction is the area capacity of a chip. Limited capacity may force the circuit onto several chips, assigning each gate to one or more of the chips. Such a mapping might lead to inefficient implementations. For example, if a path from an input to an output crosses many chip boundaries, substantial delay may result at the output. We address the problem of dividing a circuit into components so that the maximum delay at the outputs is minimized.

We consider circuit partitioning with replication of nodes, i.e., a gate may be assigned to more than one component in the layout. Following [1] and [3], we refer to each such component as a *cluster*, and to the problem as the *Circuit Clustering Problem*. Lawler, Levitt and Turner [3] gave a polynomial-time optimum solution to this problem assuming the *unit delay model* (using the terminology of [1]). In this model, no delay is associated with any gate or interconnection linking two gates within a cluster. A delay of one time unit is encountered along any interconnection crossing a cluster boundary.

The unit delay model is not very realistic, as it assumes that the inter-chip delay totally outweighs any delay within a chip. As more gates are packed onto a single chip, a path might pass through many gates, incurring a substantial delay within the chip itself.

Murgai, Brayton, and Sangiovanni-Vincentelli [1] proposed the *general delay model*, a more realistic extension of the unit delay model. In the general delay model,

1. each gate v of the network has a delay given by $\delta(v)$,

*This work was partially supported by the Texas Advanced Research Program under Grant No. 003658459 and an MCD Fellowship from the University of Texas at Austin.

2. no delay is encountered on an interconnection linking two gates in the same cluster, and

3. a delay of D time units (D is a specified constant) is encountered on every interconnection linking two gates in different clusters.

The general delay model is quite powerful, and can capture many timing constraints by simple extensions. For example, delay on the interconnections can be handled by adding a dummy gate (with zero area) to every interconnection, and assigning an appropriate delay to the dummy gate. This transformation increases the size of the network by a constant factor only. In [1], Murgai, Brayton, and Sangiovanni-Vincentelli give an algorithm (referred to as the GLLT algorithm) to cluster networks with the objective of minimizing delay under the general delay model. The GLLT algorithm (based on a greedy labeling procedure) is optimum only under specific conditions. In this paper, we present a provably optimum polynomial-time algorithm that clusters any combinational network such that the maximum delay through the network is minimized.

Section 2 presents a formal description of the problem. Our algorithm is described in detail in Section 3. In Section 4, we prove the optimality and the polynomial-time complexity of the algorithm. Section 5 discusses extensions of the algorithm for monotone clustering constraints. Section 6 presents some experimental results, and we conclude with some remarks in Section 7.

2 Problem Formulation

A combinational network can be represented as a directed acyclic graph $G = (V, E)$, where V is the set of nodes, and E is the set of directed edges. Each node in V represents a gate in the network and each edge (u, v) in E represents an interconnection between gates u and v in the network. The *fanin* of a node is the number of edges incident into it, and the *fanout* is the number of edges incident out of it. A *Primary Input* (PI) is a node with fanin 0, and a *Primary Output* (PO) is a node with fanout 0. We represent the set of PIs by $\mathcal{PI}$, and the set of POs by $\mathcal{PO}$. Each node has a *weight* and a *delay* associated with it. We denote the weight function by $w : V \rightarrow \mathbf{R}^+$ and the delay function by $\delta : V \rightarrow \mathbf{R}^+$, where $\mathbf{R}^+$ denotes the set of non-negative reals. A *cluster* is a set of nodes $U \subseteq V$ of the network.

Definition 1 *A* clustering *of a network $G = (V, E)$ is a triple (H, ϕ, Σ), where*

1. *$H = (V', E')$ is a directed acyclic graph,*

2. *ϕ is a function mapping V' to V such that*
   - *for every edge $(u', v') \in E'$, $(\phi(u'), \phi(v')) \in E$,*
   - *for every node $v' \in V'$ and edge $(u, \phi(v')) \in E$, there exists a unique $u' \in V'$ such that $\phi(u') = u$ and $(u', v') \in E'$, and*
   - *for every PO node $v \in V$, there exists a unique $v' \in V'$ such that $\phi(v') = v$, and*

3. *Σ is a partition of V'.*

Let $\Gamma = (H = (V', E'), \phi, \Sigma)$ be a clustering of G. For $v \in V$, $v' \in V'$, if $\phi(v') = v$, we call v' a *copy* of v. The set V' consists of all the copies of the nodes in V that appear in the clustering. Moreover, $v' \in V'$

is a PI (resp., PO) of a clustering (H, ϕ, Σ) if $\phi(v')$ is a PI (resp., PO) of G. It follows from the definition of ϕ that H is logically equivalent to G.

The weight (resp., delay) of any node v' in V' is the weight (resp., delay) of $\phi(v)$. The weight of any cluster $C \in \Sigma$, denoted by $W(C)$, is the sum of the weights of the nodes in C. We now present a notion of delay of a clustering of a network using the general delay model. The delay of an edge $(u', v') \in E$ is D if u and v are in different elements of Σ, and zero otherwise. The delay along a path $< v'_1, v'_2, \ldots, v'_k >$ in a clustering Γ is the sum of the delays of the gates $v'_1, v'_2, \ldots, v'_k$ and the delays on the interconnection edges $(v'_1, v'_2), \ldots, (v'_{k-1}, v'_k)$. The delay of a node v' in clustering Γ, denoted by $d_\Gamma(v')$, is the maximum delay along a path from a PI to v'. The delay of a clustering Γ is the maximum delay of a PO in Γ.

Definition 2 *Given a combinational network $G = (V, E)$ with weight function $w : V \rightarrow \mathbf{R}^+$, weight capacity M, and delay function $\delta : V \rightarrow \mathbf{R}^+$:*

1. *A clustering $\Gamma = (H, \phi, \Sigma)$ of G is* feasible *if for every cluster $C \in \Sigma$, $W(C)$ is at most M.*
2. *The* Circuit Clustering Problem *is to compute a feasible clustering Γ of G such that the delay of Γ is minimum among all feasible clusterings of G.*

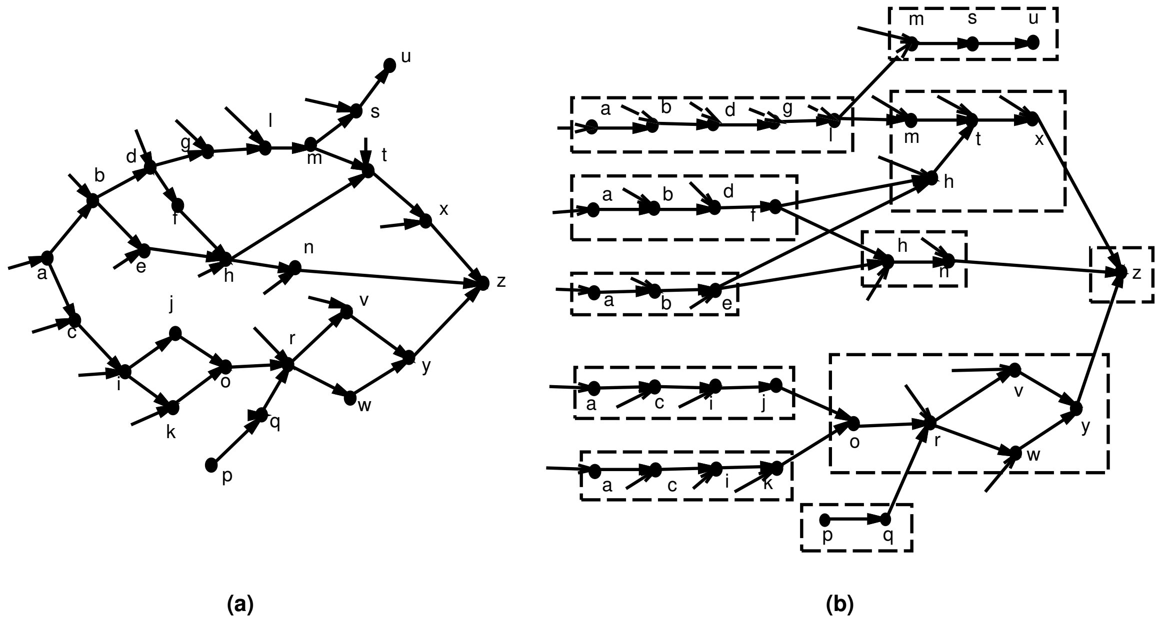

Figure 1: An instance of the Circuit Clustering Problem. (a) A circuit before clustering. (b) A clustering solution.

Figure 1 shows an instance of the Circuit Clustering Problem (taken from [3]). The combinational circuit is represented by the directed acyclic graph in Figure 1(a). An optimum clustering solution (under the unit delay model with $M = 5$) is shown in Figure 1(b).

3 An Optimum Algorithm for Delay Minimization

The algorithm consists of two phases: *Labeling* and *Clustering*. In the labeling phase, we label each node v with the maximum delay at the unique copy of v in an optimum clustering of G_v, where G_v denotes the graph consisting of v and all of its predecessors. We denote an optimum clustering of G_v by Γ_v. (Since Γ_v has a unique copy of v, we use v to denote this copy as well.) This labeling is motivated by the following:

Lemma 1 *For any node v and any clustering Γ of network G, we have $d_\Gamma(v') \geq d_{\Gamma_v}(v)$ for every copy v' of v in Γ.* ■

Lemma 1 follows from the observation that any clustering of G induces clusterings on G_v. Therefore, the maximum delay at any copy of v in a clustering Γ of G is at least the maximum delay in an optimum clustering of G_v.

The clustering phase consists of generating the clusters based on the labeling done in the labeling phase. The clustering solution thus obtained is an optimum clustering of the network.

3.1 The Labeling Phase

This phase assigns to each node v in G, a label $\ell(v)$. The network is processed in topological order. For each of the PIs v, we assign $\ell(v) = \delta(v)$. We now give a procedure to label a node v, all of whose predecessors have been labeled. For each $u \in G_v \setminus \{v\}$ we compute $\ell_v(u)$ as follows: $\ell_v(u) = \ell(u) + \Delta(u, v)$, where $\Delta(u, v)$ is the maximum delay along any path from the output of u to the output of v, ignoring delays on the interconnections. For any subset S of $G_v \setminus \{v\}$, let $m(S)$ denote any node in S with the maximum value of ℓ_v. The following algorithm computes $\ell(v)$:

> **Algorithm Labeling**(G, v);
> **begin**
> $cluster(v) \leftarrow \{v\}$; $\ell_1(v) \leftarrow l_2(v) \leftarrow 0$;
> Compute $\ell_v(u)$ for each $u \in G_v \setminus \{v\}$;
> $S \leftarrow G_v \setminus \{v\}$ sorted in non-increasing order of ℓ_v value.
> **while** $(S \neq \emptyset)$ **and** $(W(cluster(v)) + w(m(S)) \leq M)$
> $cluster(v) \leftarrow cluster(v) \cup \{u\}$;
> $S \leftarrow S \setminus m(S)$;
> **if** $cluster(v) \cap \mathcal{PI} \neq \emptyset$
> $\ell_1(v) \leftarrow max\{\ell_v(x) \mid x \in cluster(v) \cap \mathcal{PI}\}$;
> **endif**
> **if** $S \neq \emptyset$ $\ell_2(v) \leftarrow \ell_v(m(S)) + D$;
> **endif**
> $\ell(v) \leftarrow max\{\ell_1(v)\ ,\ell_2(v)\}$;
> **end**

A key idea in the labeling phase is the computation of the function ℓ_v. (Note that there is a different function ℓ_v for every v). By the definition of ℓ_v, $\ell_v(u)$ is a lower bound on the delay along any path from a primary input to v that passes through u. The greater the value of $\ell_v(u)$, the more the need to include u in $cluster(v)$. Hence we try to cluster v with as many high ℓ_v-valued nodes as the capacity constraint permits. After building $cluster(v)$, v is labeled by considering all possible paths from an input to the output of v. All of the paths can be divided into two categories:

(i) Paths that lie entirely in $cluster(v)$. Such paths start from a primary input that is in $cluster(v)$, and never exit the cluster. The maximum delay along any such path is

$$\ell_1(v) = max\{\ell_v(u) \mid u \in cluster(v) \cap \mathcal{PI}\} \tag{1}$$

(ii) Paths that cross the "boundary" of $cluster(v)$. Among these paths, the maximum delay is

$$\ell_2(v) = max\{\ell_v(u) + D \mid u \in G_v \setminus cluster(v)\} \tag{2}$$

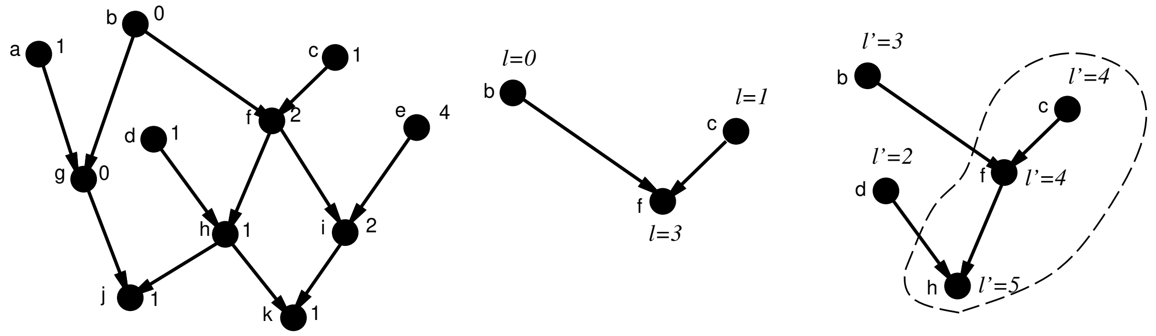

Figure 2: The Labeling Phase. (a) The graph G representing the circuit *c17*. (b) Labels of nodes b, c, and f. (c) Computing the label of h.

Thus the label $\ell(v)$ is given by

$$\ell(v) = max\{\ell_1(v), \ell_2(v)\} \tag{3}$$

Consider the example in Figure 2(a). The graph G represents the ISCAS85 example circuit $c17$. There are 11 nodes, named a through k. The 5 PIs are a, b, c, d, and e. There are 2 POs, j and k. Each node has a weight of 1. The delays on the nodes are specified in the figure. Let the capacity M of a cluster be 3, and the inter-cluster delay D be 3. **Labeling** first assigns labels to the PI nodes, equal to their respective delays. The label on node f is easily seen to be 3 (Figure 2(b)). We now wish to label node h. We obtain the subgraph G_h and compute the function ℓ_h (denoted ℓ' in Figure 2(c)) for each node in G_h: $\ell_h(b) = 0 + 3 = 3$; $\ell_h(c) = 1 + 3 = 4$; $\ell_h(d) = 1 + 1 = 2$; and $\ell_h(f) = 3 + 1 = 4$. Since the cluster capacity is 3, only nodes f and c can be accomodated into $cluster(v)$. We observe that although both c and d have the same label, c is a more critical node for f due to a path with greater delay. Note that while labeling a node v, we use the labels of all the predecessors of v in the network. Moreover, unlike in a greedy algorithm, we do not make use of the clusters computed in any previous iteration. Instead $cluster(v)$ for a node v is computed by considering the entire subgraph G_v.

3.2 The Clustering Phase

The labeling phase generates a cluster for each node of G. The clustering phase computes a clustering of G by selecting clusters among those generated in the first phase. For any cluster C, let $I(C)$ denote the set of inputs to C. We maintain a list L of nodes whose clusters will be selected and a set S of selected clusters. Initially L is set to the set of primary outputs of G. The following three steps are repeated until L is empty: (i) remove node u from L and add $cluster(u)$ to S, (ii) compute $I(cluster(u))$, and (iii) for every node $x \notin L$ such that $x \in I(cluster(u))$, if $cluster(x) \notin S$, add x to L. (Note that for every input y of any cluster in S, $cluster(y)$ is selected in this phase.)

A clustering $\Gamma = (H, \phi, \Sigma)$ can be easily defined from S and I. The vertices of H are the different copies of the vertices in V. For every vertex $v \in V$, we have a copy of v in H for every cluster in S that v belongs to. The function ϕ maps every copy v' of $v \in V$ to v. The edges in the clustering are of two kinds: edges within clusters and edges between clusters. The edges within any cluster C are all the edges in the subgraph of G induced by the vertices of C. The edges from cluster $C = cluster(u)$ to C' are all the edges from the copy of u in C to all the neighbors of u in C'. The set Σ consists of all the subsets of V' that correspond to the clusters in S.

For the example of Figure 2, the final clustered circuit is given in Figure 3. We start with the POs j and k and select clusters $cluster(j)$ and $cluster(k)$ respectively. The set of nodes that form inputs to these clusters is $\{c, d, f, g, h\}$. So clusters rooted at these nodes are selected. Finally, $cluster(b)$ is selected since b is an input to $cluster(h)$. This optimum clustering solution requires replication of nodes b, c, f, and h.

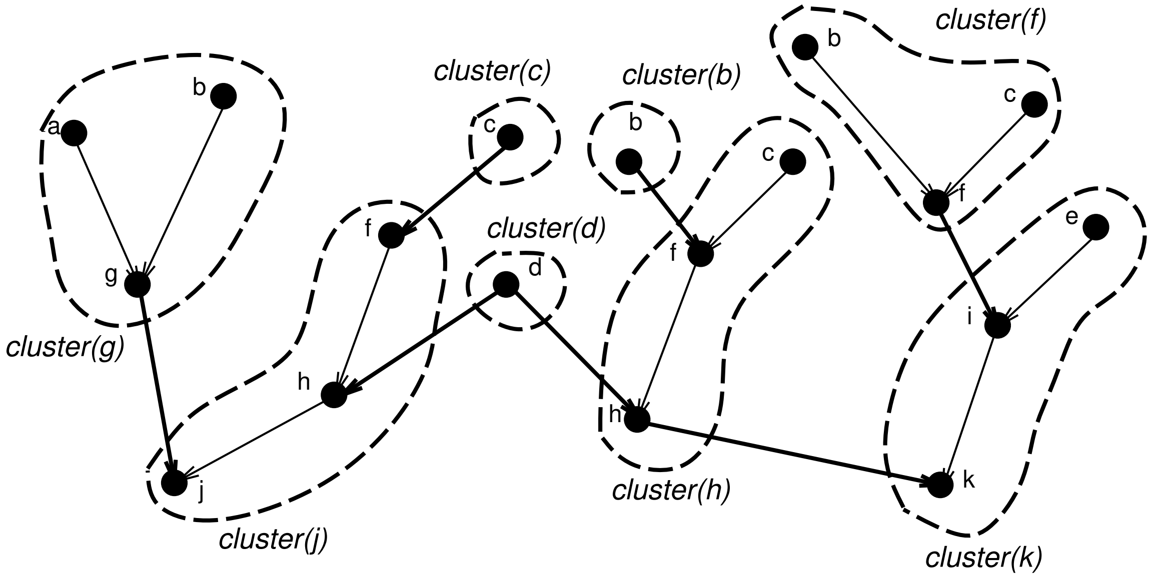

Figure 3: An optimum clustering of the circuit *c17*.

The complete algorithm, named **Clustering**, includes both the labeling and clustering phases and is summarized in the following:

Algorithm Clustering(G, w, δ, D, M)
Output: Clustering Γ.
begin
Compute the maximum delay matrix Δ;
for each PI i, **do** $\ell(i) \leftarrow \delta(i)$;
Sort the non-PI nodes of G in topological order to obtain list T;
while T is non-empty
    Remove the first node v from T;
    Compute G_v;

for each node $u \in G_v \setminus \{v\}$ **do**

$\ell'(u) \leftarrow \ell(u) + \Delta(u, v)$;

Sort the nodes in $G_v \setminus \{v\}$ in order of decreasing value of ℓ_v to form list P;

Call **Labeling**(v);

endwhile

$L \leftarrow \mathcal{PO}$;

$S \leftarrow \emptyset$;

while L is not empty

Remove a node v from L;

$S \leftarrow S \cup \{cluster(v)\}$;

$I(cluster(v)) = \{x \in V \mid x \text{ is an input to } cluster(v)\}$;

$L \leftarrow L \cup (I(cluster(v)) \setminus \{y \mid cluster(y) \in S\})$;

endwhile

Generate clustering Γ from S and function I;

end

4 Proof of Optimality

Theorem 1 *For any directed acyclic combinational network $G = (V, E)$ with weight function w, delay function δ, inter-cluster delay D, and maximum cluster capacity M,* **Clustering** *computes an optimum clustering* Γ.

We prove Theorem 1 by establishing the correctness of the labeling and clustering phases. Lemma 2 verifies the labeling phase.

Lemma 2 *For every $v \in V$, we have $\ell(v) \leq d_{\Gamma_v}(v)$.*

Proof: The proof is by induction on the topological ordering of V.

Induction Basis : For each PI i, **Labeling** assigns $\ell(i) = \delta(i)$, which is the maximum delay in the optimum clustering of $G_i = \{i\}$.

Induction Step : Assume the statement is true for all predecessors of v, i.e., $\ell(u) \leq d_{\Gamma_u}(u)$ for all $u \in G_v \setminus \{v\}$. Based on Equations (1) and (2) in Section 3.1, we consider the following two cases :

Case 1: $\ell(v) = \ell(u) + \Delta(u, v)$, for some $u \in cluster(v) \cap \mathcal{PI}$. Since in this case $\ell(u) = \delta(u)$, $\ell(v)$ is just the delay along the path from u to v. Clearly the maximum delay at the output of v will be at least $\ell(v)$.

Case 2 : $\ell(v) = \ell(u) + \Delta(u, v) + D$, for some $u \notin cluster(v)$. We first observe that for every $x \in cluster(v) \setminus \{v\}$, $\ell(v) \leq \ell(x) + \Delta(x, v) + D$. Therefore, for $y \in cluster(v) \setminus \{v\}) \cup \{u\}$, we have

$$\ell(v) \leq \ell(y) + \Delta(y, v) + D \tag{4}$$

Now consider Γ_v, an optimum clustering of G_v. Consider the cluster C that contains the unique copy of v, which we denote by v for convenience. Our proof is by contradiction. Assume $d_{\Gamma_v}(v) < \ell(v)$. Let

$X = \{x \notin cluster(v) \mid \ell(v) = \ell(x) + \Delta(x, v)\}$. Hence, X is the set of nodes not in $cluster(v)$ that have the maximum value of ℓ_v. Since u belongs to X, X is non-empty.

If $cluster(v) \cup X \subseteq C$, then the weight of C is greater than M, violating the capacity constraints, as otherwise in the labeling phase we would have added some node in X to $cluster(v)$. Therefore, there exists $x \in cluster(v) \cup X$ such that no copy of x is in C. Let x' be any copy of x in Γ_v. Thus, $d_{\Gamma_v}(v) \geq d_{\Gamma_v}(x') + \Delta(x, v) + D$. It follows that $\ell(v) > d_{\Gamma_v}(x') + \Delta(x, v) + D$. If $x \in cluster(v)$, by Equation 4, we have $\ell(x) > d_{\Gamma_v}(x')$; otherwise $x \in X$ and we have $\ell(v) = \ell(x) + \Delta(x, v)$ implying that $\ell(x) > d_{\Gamma_v}(x')$. Since Γ_v induces a clustering of G_x, by invoking Lemma 1, we obtain that $\ell(x) > d_{\Gamma_x}(x)$, a contradiction of our induction hypothesis. Thus we have $\ell(v) \leq d_{\Gamma_v}(v)$. ■

In Lemma 3 we show that the clustering solution generated in the clustering phase is an optimum solution. Let Γ denote the clustering output by **Clustering**. If $cluster(v)$ is present in Γ, we denote the particular copy of v in $cluster(v)$ by v.

Lemma 3 *For every node $v \in V$, if $cluster(v)$ is in Γ, then $d_\Gamma(v) = \ell(v)$.*

Proof: Let v be any vertex such that $cluster(v)$ is in Γ. By Lemma 1 and 2 we have $d_\Gamma(v) \geq d_{\Gamma_v}(v) \geq \ell(v)$. The proof in the other direction, $d_\Gamma(v) \leq \ell(v)$, is by induction on the topological ordering of V.

Induction Basis : The maximum delay at the output of a PI i in any clustering solution is $\delta(i)$. Since **Labeling** assigns $\ell(i) = \delta(i)$, the statement holds.

Induction Step : Assume $d_\Gamma(x) \leq \ell(x)$ for every $x \in G_v \setminus \{v\}$ such that $cluster(x)$ is in Γ. If a cluster is generated for v, $d_\Gamma(v)$ is given by the maximum of

$$\max\{d_\Gamma(u) + \Delta(u, v) \mid u \in cluster(v) \cap \mathcal{PI}\}, \text{ and}$$
$$\max\{d_\Gamma(u') + \Delta(u, v) + D \mid u' \text{ is a copy of } u, \text{ an input node to } cluster(v)\}.$$

If $d_\Gamma(v) = d_\Gamma(u) + \Delta(u, v)$ for some PI u, then $d_\Gamma(v) = \ell(v)$ (follows from Equation (1) of Section 3.1); otherwise $d_\Gamma(v) = d_\Gamma(u') + \Delta(u, v) + D$ from some u' that is a copy of u, an input to $cluster(v)$. In this case, **Clustering** also generates $cluster(u)$. Since u is a predecessor of v, by the induction hypothesis, we have $d_\Gamma(u) = \ell(u)$. So $d_\Gamma(v) = \ell(u) + \Delta(u, v) + D \leq \ell(v)$. ■

Proof of Theorem 1: By Lemma 3, for every PO node u, we have $d_\Gamma(u) = \ell(u)$. Since we have $\ell(u) = d_{\Gamma_u}(u)$ by Lemma 2, it follows that $d_\Gamma(u) = d_{\Gamma_u}(u)$. ■

Theorem 2 *For a combinational network $G = (V, E)$,* **Clustering** *runs in $O(n^2 \log n + nm)$ time, where $n = |V|$ and $m = |E|$.*

Proof: The computation of the matrix Δ can be reduced to an all-pairs shortest path problem by a suitable transformation of the graph. The all-pairs shortest path problem for directed acyclic graphs can be solved in $O(n(n + m))$ time [4]. Sorting the nodes in topological order takes $O(n + m)$ time. For each node v, G_v can be constructed in $O(n + m)$ time, and the nodes can be sorted according to the values of ℓ' in $O(n \log n)$ time. So the complexity of the first while loop is $O(n^2 \log n + nm)$. The generation of the clustering can be done in $O(n(n + m))$ time. Hence the complexity of the entire algorithm is $O(n^2 \log n + nm)$. ■

5 Delay Optimization with Monotone Clustering Constraints

Our algorithm can be easily generalized to compute the optimum clustering solution under any monotone clustering constraint. A clustering constraint is *monotone* if and only if any connected subset of nodes in a feasible cluster is also feasible [3]. Clearly the capacity constraint is a monotone clustering constraint.

To solve the delay optimization problem under any monotone constraint, the labeling phase has to be changed to test for the feasibility of the cluster being computed, under the particluar clustering constraint. The correctness follows from the observation that the partial cluster obtained at any step during the computation of $cluster(v)$ for any node v by the labeling phase, is always a connected subset of $cluster(v)$. The proofs of all the theorems and lemmas in Section 5 follow with minor modifications. Thus we have the following theorem

Theorem 3 *The Circuit Clustering Problem for delay minimization can be solved optimally under any monotone clustering constraint in polynomial time in the size of the circuit.* ■

One implication of this is that the problem of clustering with pin limitations can be solved in polynomial time for tree networks. The "pin limitation" constraint is the restriction on the number of signals that cross a cluster. Since such a constraint is a monotone constraint for both rooted and nonrooted trees [3], our **Clustering** algorithm, with appropriate modifications can compute an optimum clustering solution. Unfortunately, for a general combinational network, the pin constraint is not monotone. Cong and Ding [5] have given an optimum algorithm for a special case of clustering under pin constraints in the unit delay model, that has applications in FPGA designs.

6 Experimental Results

We have implemented the Clustering algorithm in C on the SUN SPARC workstations. The algorithm was tested on some ISCAS combinational networks. Results for five ISCAS circuits are shown in Table 6. No further logic minimization was done on these circuits. For our experiments, we chose: $\delta(v) = 1$, $w(v) = 1$ for all gates v. We set the cluster capacity M to 100, and the cluster-interconnection delay D to 2.

Example Network	Number of Nodes	Maximum Delay	Total Time in seconds
c432	196	15	1.5
c499	243	11	2.0
c880	443	17	4.0
c1355	587	24	20.7
c1908	913	39	37.1

Table 1: The performance of Algorithm **Clustering**

The table shows the number of nodes in the network[1], the maximum delay through the network (as

[1]The ISCAS '85 format, in which the networks were specified, lists fanout branches separately as distinct nodes. For our experiments, we have not added nodes for any fanout branch.

calculated by **Clustering**), and the total time taken in seconds by the algorithm.

7 Concluding Remarks

We have presented a polynomial time optimum algorithm for the problem of clustering networks to minimize delay, subject to capacity constraints, under the general delay model. Since the general delay model can be very easily extended to handle delay on all interconnections and arbitrary arrival times at the primary inputs, our method applies to the most general clustering problem. Moreover, the algorithm can be generalized for any monotone clustering constraint.

Our algorithm does not guarantee the minimum number of clusters in the solution with optimum delay. We can overcome this to some extent by using techniques mentioned in [1], in a postprocessing phase to reduce the number of nodes and clusters, without changing the delay through the circuit.

References

[1] R. Murgai, R.K. Brayton and A. Sangiovanni-Vincentelli, "On Clustering for Minimum Delay/Area", *Proceedings of the IEEE International Conference on Computer-Aided Design*, November 1991, pages 6-9.

[2] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley-Tuebner, 1990.

[3] E.L. Lawler, K.N. Levitt and J. Turner, "Module Clustering to Minimize Delay in Digital Networks", *IEEE Transactions on Computers*, Vol. C-18, No. 1, January 1966, pages 47-57.

[4] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms*, McGraw-Hill 1990.

[5] J. Cong and Y. Ding, "An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA designs", *Proceedings of the IEEE International Conference on Computer-Aided Design*, November 1992, pages 48-53.