Optimal Wire Spacing Under Cross Talk Constraints

Paul B. Morton
Wayne Dai

UCSC-CRL-98-09
August 3, 1998

Jack Baskin School of Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064 USA

ABSTRACT

As VLSI technology continues to advance, the spacing between adjacent wires continues to decrease, causing a corresponding increase in capacitive coupling between adjacent wires. This trend makes it increasingly likely that we will generate routings which violate one or more of the noise margins on the net's sink pins. At the same time, these technologies are becoming more routing limited. These two trends are increasingly forcing physical designers to use scarce routing resources to decrease the coupling between adjacent wires. From this we can see that what is needed is a set of tools, applicable in an area routing environment, that will generate routings that meet all the noise constraints while sacrificing the minimum amount of routing resources. In this paper we will present a polynomial time algorithm, which can be used in an area routing environment, for determining the optimal spacing between a victim net and it's adjacent aggressor nets such that none of the noise margins of the victim net's sink pins are violated, while consuming the minimum amount of routing resources.

Keywords: cross talk, constrained routing, wire spacing, noise margin.

1 Introduction

As VLSI technology continues to advance, the spacing between adjacent wires continues to decrease, causing a corresponding increase in capacitive coupling between adjacent wires. The increased coupling has, in turn, lead to an increase in the amount of cross talk noise being coupled into a given net. This trend makes it increasingly likely that we will generate routings which violate one or more of the noise margins on the net's sink pins. At the same time, these technologies are becoming more routing limited, as can be seen by the ever increasing number of routing layers. These two trends are increasingly forcing physical designers to use scarce routing resources, in the form of increased spacing, to to decrease the coupling between adjacent wires. From this we can see that what is needed is a set of tools that will generate routings that meet all the noise constraints while sacrificing the minimum amount of routing resources. Further, it is clear, for technologies having more than two routing layers, that any tools developed to help solve this problem need to be applicable in an "over the cell", or area routing environment.

Over the past few years there have been several proposed strategies for minimizing cross talk noise during routing. These include [1], and [2] for chip level routing, and [3], [4], and [5] for MCM level routing. However, since none of these strategies directly conserve routing resources, and all of them are based on channel or switch box routers, they cannot be used to solve our problem. An additional notable work in this area is [6] which proposes a strategy for meeting cross talk constraints while using a minimum amount of routing resources, however, in this case the technique is only applicable to channel routers, and they make the very conservative assumption that each net has a single cross talk noise margin.

In this paper we will present a polynomial time algorithm, which can be used in an area routing environment, for determining the optimal spacing between a victim net and it's adjacent aggressor nets such that none of the noise margins of the victim net's sink pins are violated, while consuming the minimum amount of routing resources. Over the past few years there has been a great deal of work done on the related problem of determining the optimal sizing and spacing to meet timing constraints, see for example [7], [8], [9], and [10], however, none of these have directly or adequately addressed the cross talk noise problem.

This new spacing algorithm will be particularly powerful when used in combination with a flexible area router such as SURF [11], where the victim net's adjacency information can be quickly and accurately estimated during the "rubber band" routing phase. The spacing information generated by the new spacing algorithm can then be used by SURF to construct the final rectilinear routing with a set of spacings that will eliminate the cross talk noise problem.

Fundamental to the work done in this paper is the work done by Devgan ,in [12], which gives us a closed form expression that allows us to quickly and accurately calculate the maximum cross talk noise seen at each of the sink pins of a victim net. This noise metric is far more accurate than the simple "coupling length" based noise metrics currently being used in layout synthesis, and is far less expensive than computing the cross talk noise using parasitic extraction and circuit simulations. We will use this closed form expression in a dynamic programming strategy, similar to that used in [10] and [7], to develop a polynomial time algorithm to solve the discrete spacing version of the optimal wire spacing under cross talk constraints problem outlined above. The major contributions of this paper include:

- The formulation of the optimal wire spacing under cross talk constraints problem.

- A polynomial time solution to the discrete spacing version of the optimal wire spacing under cross talk constraints problem.
- Incremental dominance pruning, as well as several heuristic pruning strategies used by the dynamic program.
- A proof showing why the dominance property can be used as a pruning strategy in our dynamic program.

2 Problem formulation

Given a routed victim net an a set of routed aggressor nets, we would like to determine a corresponding set of spacings, which, if used to separate the aggressor nets from the victim net, would meet the cross talk noise margin constraints at each sink pin of the victim net, and would minimize the amount of routing resource used to meet these constraints. To accomplish this, we need to compute, for a given set of spacings, the cross talk noise induced on each sink pin of the victim net. Further, we need to compute the total routing area consumed by the spacing between the victim net and each of the aggressor nets.

From [12] we can determine the maximum cross talk noise voltage on any node of a victim net as

$$V_n = V_{Par(n)} + R_n \sum_{\substack{i \in Des(n) \\ j \in Ag(i)}} C_{ij} \dot{U}_j \tag{1}$$

where

- $Des(n)$ is the set of victim net nodes that are descendants of node n.
- $Par(n)$ is the parent node of node n.
- $Ag(i)$ is the set of aggressor nets that are coupled into node i of the victim net.
- V_n is the maximum cross talk noise voltage, induced on node n, by all down stream aggressor nets.
- R_n is the resistance connecting node n to node $Par(n)$.
- $\dot{U}_j$ is the slope of the aggressor net signal on aggressor net j.
- C_{ij} is the coupling capacitance between aggressor net j and the segment connecting node i to node $Par(i)$.

To illustrate, consider the routing depicted in Fig. 1, containing four nodes, N_0 through N_3, and six aggressor nets, Ag_1 through Ag_6. This can be reduced to the network shown in Fig. 2, where R_0 represents the net's source resistance. Using (1) we can see that

$$\begin{aligned}
V_0 &= R_0 \left(C_{11}\dot{U}_1 + C_{25}\dot{U}_5 + C_{26}\dot{U}_6 + C_{32}\dot{U}_2 + C_{33}\dot{U}_3 + C_{34}\dot{U}_4\right) \\
V_1 &= R_1 \left(C_{11}\dot{U}_1 + C_{25}\dot{U}_5 + C_{26}\dot{U}_6 + C_{32}\dot{U}_2 + C_{33}\dot{U}_3 + C_{34}\dot{U}_4\right) + V_0 \\
V_2 &= R_2 \left(C_{25}\dot{U}_5 + C_{26}\dot{U}_6\right) + V_1 \\
V_3 &= R_3 \left(C_{32}\dot{U}_2 + C_{33}\dot{U}_3 + C_{34}\dot{U}_4\right) + V_1
\end{aligned}$$

Since the analysis in [12] assumes that all aggressor net signals are ramps, we have

$$\dot{U}_j = 0.8 \frac{V_{DD}}{t_j} \tag{2}$$

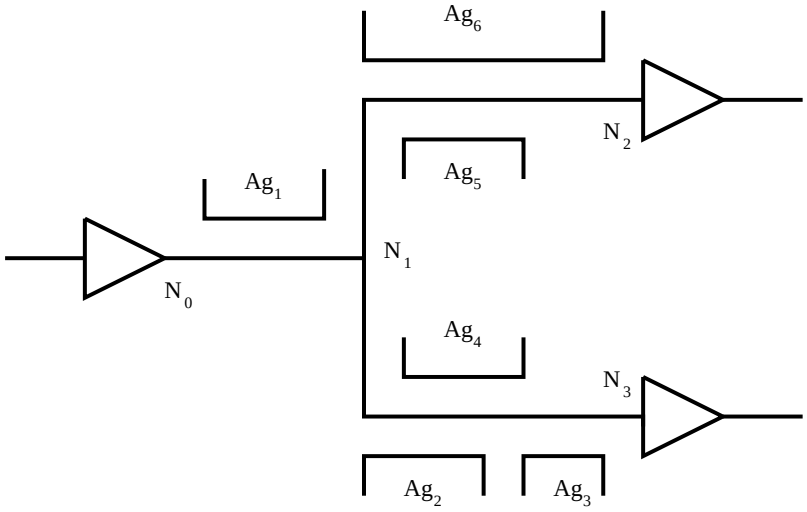

Figure 1: Victim net with six adjacent aggressor nets.

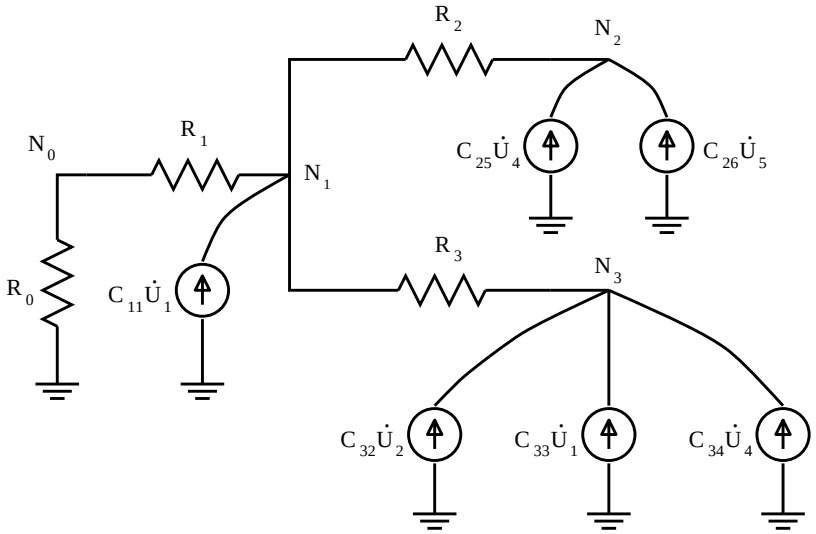

Figure 2: Electrical network for a victim net with six adjacent aggressor nets.

where V_{DD} is the supply voltage and t_j is the 10% to 90% rise time of aggressor net j. Further, we have that

$$C_{ij} = C_0 \frac{L_{ij}}{S_{ij}} \tag{3}$$

where

- L_{ij} is the length of the adjacency between aggressor net j and the segment connecting node i to node $Par(i)$.
- S_{ij} is the distance separating the adjacent segments.
- C_0 is a proportionality constant.

Substituting (2) and (3) into (1) we have

$$V_n = V_{Par(n)} + 0.8\, V_{DD}\, C_0\, R_n \sum_{\substack{i \in Des(n) \\ j \in Ag(i)}} \frac{L_{ij}}{S_{ij}\, t_j} \tag{4}$$

The routing area consumed by a set of spacings is

$$A = \sum_{\substack{i \in N \\ j \in Ag(i)}} L_{ij}\, S_{ij} \tag{5}$$

where N is the set of all victim net nodes.

Using (4) and (5) we can formulated the optimal wire spacing under cross talk constraints problem as the following constrained minimization problem

$$min \sum_{\substack{i \in N \\ j \in Ag(i)}} L_{ij}\, S_{ij} \tag{6}$$

subject to

$$\begin{aligned} V_n &\leq M_n \qquad \forall\ n \in N \\ S_{ij} &\geq S_{min} \qquad \forall\ i \in N,\ j \in Ag(i) \end{aligned} \tag{7}$$

from which we can determine an optimal set of spacings, $\{S_{ij} \mid \forall\ i \in N,\ j \in Ag(i)\}$, where M_n is the noise margin for node n of the victim net, and S_{min} is the minimum allowable spacing between two adjacent segments. Note that $M_n = \infty$ for victim net nodes that do not contain a sink pin.

Assuming we are routing on a uniform routing grid, L_{ij} and S_{ij} can be restricted to integer values which represent the number of grids over which the adjacency occurs and the number of grids separating the adjacent segments, respectively. Assuming that each S_{ij} is selected from a finite set of spacings, $\{1, 2, \ldots, S_{max}\}$, then the general optimization problem defined by (6) and (7) can be reduced to the following combinatorial optimization problem

$$min \sum_{\substack{i \in N \\ j \in Ag(i)}} L_{ij}\, S_{ij} \tag{8}$$

subject to

$$\begin{aligned} V_n &\leq M_n \qquad \forall\ n \in N \\ S_{ij} &\in \{1, 2, \ldots, S_{max}\} \qquad \forall\ i \in N,\ j \in Ag(i) \end{aligned} \tag{9}$$

where we want to determine the set of spacings $\{S_{ij} \mid \forall\ i \in N,\ j \in Ag(i)\}$

3 Dynamic programming algorithm

In this paper, we will use a dynamic programming approach, similar to [10], to approximate the values of S_{ij} that satisfy (8) and (9). In order to simplify the development of the dynamic program, we will assume that we have a binary network, with at most one adjacency per node. Each node, n, in the binary network contains the following information:

- R – The resistance connecting node n to node $Par(n)$.
- L – The length of the adjacency associated with node n.
- t – The rise time of the signal on the adjacent aggressor net.
- M – The noise margin associated with node n.
- P_L – The pointer to node n's left child.
- P_R – The pointer to node n's right child.

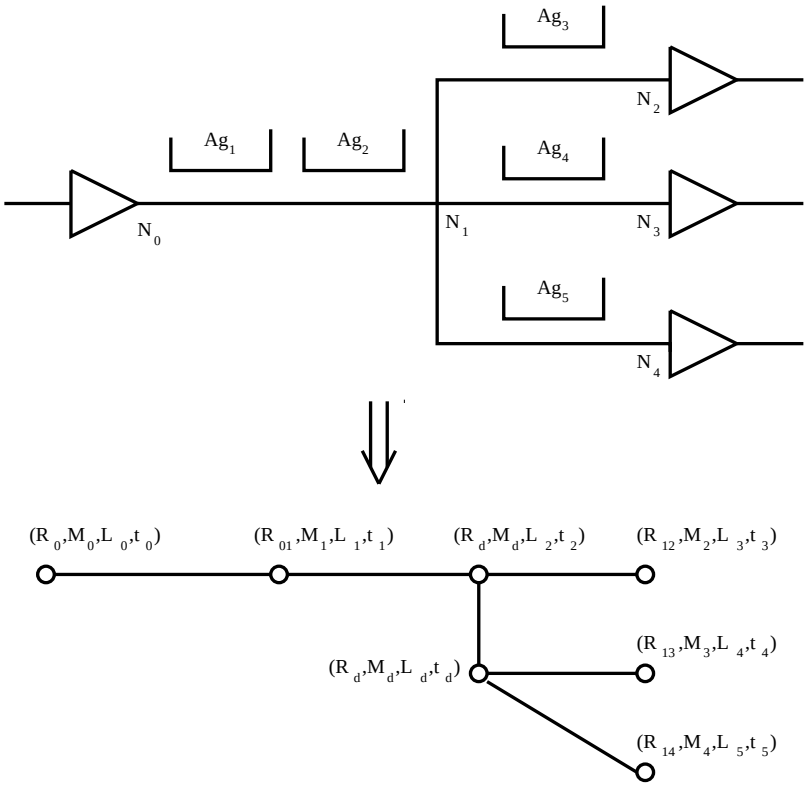

Figure 3: General network to binary network conversion.

To convert a general network with multiple adjacencies per node to a binary network with at most one adjacency per node we introduce intermediate nodes with zero resistance and infinite noise margin, as illustrated in Fig. 3, where

- R_0 is the nets source resistance.
- $R_d = 0$
- R_{01}, R_{12}, R_{13}, and R_{14} are the resistances between nodes N_0 and N_1, N_1 and N_2, N_1 and N_3, and N_1 and N_4, respectively.
- $M_d = \infty$
- M_1 through M_4 are the noise margins at nodes N_1 through N_4, respectively.
- $L_d = 0$
- L_1 through L_5 are the adjacency lengths associated with Ag_1 through Ag_5, respectively.
- $t_d = \infty$
- t_1 through t_5 are the rise times associated with Ag_1 through Ag_5, respectively.

In order to implement a dynamic programming solution we need to augment each of the binary network nodes with a set of "partial solutions". A partial solution set, for a node, n, is the set of feasible solutions to (8) and (9) when they are applied to the subnetwork rooted at node n. Note that for the root node, a partial solution will be called a solution, and a partial solution set will be called a solution set.

Each element of a partial solution set for a node, n, consists of a six touple (A, I, Mr, S, P_L, P_R), where

- A is the total area consumed by this partial solution.
- I is the total current drawn through node n's resistor.
- Mr is the "minimum remaining" noise margin.
- S is the spacing being used at node n of this partial solution.
- P_L and P_R are pointers to a partial solution from the left and right children, respectively, on which this partial solution is based.

Specifically, consider the general situation depicted in Fig. 4 where we have a binary network node, n, with two children. Each child has a partial solution set, PSS_L and PSS_R, from which we can choose partial solutions $PS_L = (A_L, I_L, Mr_L, S_L, P_{LL}, P_{RL})$, and $PS_R = (A_R, I_R, Mr_R, S_R, P_{LR}, P_{RR})$ respectively. Node n is connected to its parent through resistance R_n, and has a noise margin M_n. Further, from (2) and (3), the current injected into node n is

$$I(S, L_n, t_n) = 0.8 C_0 V_{DD} \frac{L_n}{S\, t_n}$$

We compute the partial solution, $PS = (A, I, Mr, S, P_L, P_R)$, for node n as follows

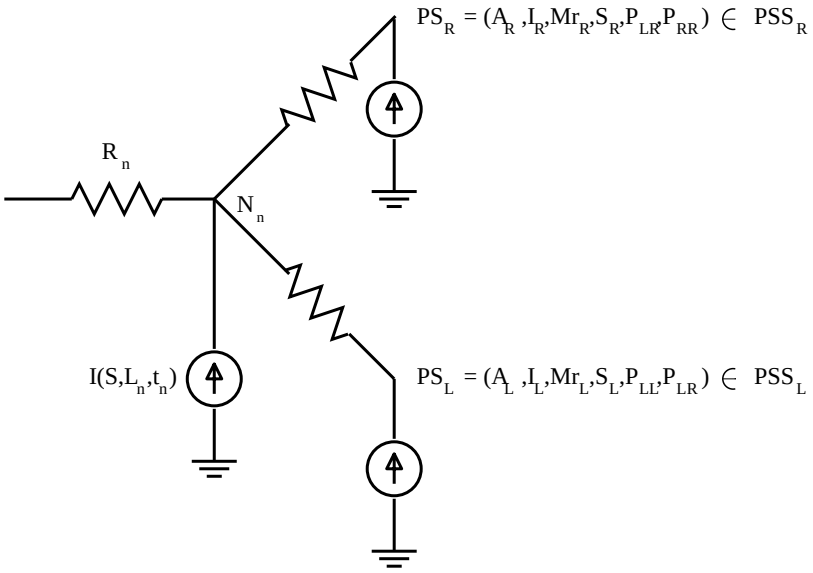

Figure 4: General binary network node.

$$A = A_L + A_R + S\, L_n \tag{10}$$

$$I = I_L + I_R + I(S, L_n, t_n) \tag{11}$$

$$Mr = min\{Mr_L, Mr_R, M_n\} - I\, R_n \tag{12}$$

and P_L and P_R point to PS_L and PS_R, respectively. We generate the entire partial solution set for node n by computing a partial solution for all combinations of $PS_L \in PSS_L$, $PS_R \in PSS_R$, and $S \in \{1, 2, \ldots S_{max}\}$. Note that for a node that has no left child we would use $A_L = 0$, $I_L = 0$, and $Mr_L = \infty$ to compute (10), (11), and (12). The same reasoning can be applied to a node with no right child.

4 Solution set size

To determine a set of optimal spacings we need to compute the partial solution set for each node in the binary network, starting at the leaves and working back to the root. Once the solution set has been generated for the root node we select the one with the smallest area. To enumerate the set of spacings that compose this solution, we traverse the binary tree formed by the solution's left and right points, P_L and P_R.

To determine a bound on the size of any partial solution set we need to consider the maximum number of unique values that each of the A, I, and Mr components of a partial solution can take on.

To begin, let us consider the area component of a partial solution. Since we are working with nets that have been routed on a uniform routing grid, we know that the total net length, L_{net}, can be represented as

$$L_{net} = l_{net}\delta_{grid}$$

where l_{net} is an integer and δ_{grid} is the grid spacing. Because of this, there can be at most $2l_{net}S_{max}$ unique areas associated with any partial solution set.

Now let us consider the current component of a partial solution. Due to the rise time and the form of the expression to compute the current associated with each adjacency, we find that the set of current values associated with a partial solution set can only be bounded by ${S_{max}}^{N_A}$, where N_A is the number of aggressor nets adjacent to the victim net. This is clearly an unmanageable number of possibilities. To resolve this problem we map the computed value for each current source to a fixed size set of evenly spaced current values. The size of this set is I_{max}. Using this strategy, there can be at most $I_{max}N_A$ unique current values associated with any partial solution set. It should be noted that in order to obtain this "reasonable" upper bound on the number of unique current values, our solution will only approximate a a solution to (8) and (9).

Finally, let us consider the remaining noise margin component. In a manner similar to the current component, this component is also bounded by an exponential number of unique values, however, we need only consider one of these values for each unique paring of A and I. Specifically, for each unique pair of (A, I), for a partial solution set, we need only keep track of the (A, I, Mr) paring with the largest remaining noise margin value. All other pairings are clearly suboptimal.

Given the bounds on each component we find that the partial solution set size can be polynomialy bounded by

$$O(l_{net}S_{max}I_{max}N_A) \tag{13}$$

5 Pruning strategies

While the size of the partial solution set can be polynomialy bounded, it is clear, for any reasonable values of S_{max}, l_{net}, I_{max}, and N_A, that the size of the partial solution sets can still become unmanageably large. To combat the growth of the partial solution sets, we employ several pruning strategies. These pruning strategies are designed to identify and eliminate "dead end" partial solutions as early as possible. At present we employ four strategies

- Remaining-noise-margin prune
- Dominance prune
- Resistance-to-root prune
- Maximum-area prune

5.1 Remaining-noise-margin prune

The simplest and most obvious strategy is the "remaining-noise-margin prune". For this strategy we eliminate any obviously infeasible partial solutions, that is, any partial solutions where $Mr < 0$. It is clear form (12) that if we have a partial solution with $Mr < 0$, then any partial solutions based on this will also have an $Mr < 0$ and cannot generate a feasible solution at the root.

5.2 Dominance prune

The most powerful, and the most complex, strategy is the "dominance prune". Using this strategy we can easily generate pruned sets that are $\frac{1}{10}$ to $\frac{1}{10000}$ the size of their unpruned counterparts. The dominance prune strategy is base on a dominance relation.

Definition 1: *For two partial solutions, PS_1 and PS_2, in a partial solution set, PSS, we say that PS_1 dominates PS_2 ($PS_1 \succ PS_2$) if*

$$A_2 \geq A_1 \tag{14}$$
$$I_2 \geq I_1 \tag{15}$$
$$Mr_2 \leq Mr_1 \tag{16}$$

Using this property we can show, from Theorem 1 and Theorem 2, that any dominated partial solution of a partial solution set will lead to an infeasible or suboptimal solution in the root's solution set. Because of this we can eliminate all dominated partial solutions from a partial solution set.

Theorem 1: *Given a partial solution set ,PSS_n, for node n, and two partial solutions, $PS_{1n} \in PSS_n$ and $PS_{2n} \in PSS_n$, where $PS_{1n} \succ PS_{2n}$, then any partial solution in $PSS_{Par(n)}$ derived from PS_{2n} will be dominated by at least one other partial solution in $PSS_{Par(n)}$.*

Proof: Assume, without loss of generality, that n is the right child of $Par(n)$. Further, since the partial solutions of $Par(n)$ are formed by combining the partial solutions of $Par(n)$'s children, let us assume that PS_L is a partial solution from $Par(n)$'s left child. Let $PS_{1Par(n)}$ and $PS_{2Par(n)}$ be the partial solutions of $PSS_{Par(n)}$ formed by combining PS_{1n} with PS_L and PS_{2n} with PS_L, respectively. From (10), (11), and (12) we see

$$\begin{aligned} A_{1Par(n)} &= A_L + A_{1n} + S_n L_n \\ I_{1Par(n)} &= I_L + I_{1n} + I(S_n, L_n, t_n) \\ Mr_{1Par(n)} &= min\{Mr_L, Mr_{1n}, M_n\} - R_n I_{1Par(n)} \end{aligned}$$

and

$$\begin{aligned} A_{2Par(n)} &= A_L + A_{2n} + S_n L_n \\ I_{2Par(n)} &= I_L + I_{2n} + I(S_n, L_n, t_n) \\ Mr_{2Par(n)} &= min\{Mr_L, Mr_{2n}, M_n\} - R_n I_{1Par(n)} \end{aligned}$$

Since $PS_{1n} \succ PS_{2n}$ *then* $A_{1n} \leq A_{2n}$ *and thus*

$$A_{1Par(n)} \leq A_{2Par(n)} \tag{17}$$

Similarly, since $I_{1n} \leq I_{2n}$,

$$I_{1Par(n)} \leq I_{2Par(n)} \tag{18}$$

Finally, since $Mr_{1n} \geq Mr_{2n}$ *and* $I_{1Par(n)} \leq I_{2Par(n)}$, *then*

$$min\{Mr_L, Mr_{1n}, M_n\} \geq min\{Mr_L, Mr_{2n}, M_n\}$$

and $R_n I_{1Par(n)} \leq R_n I_{2Par(n)}$, *thus*

$$Mr_{1Par(n)} \geq Mr_{2Par(n)} \tag{19}$$

From (17), (18), and (19) we see that

$$PS_{1Par(n)} \succ PS_{2Par(n)}$$

and thus there is at least one partial solution in $PSS_{Par(n)}$ *which dominates the partial solution derived from* PS_{2n}.

Note that it is possible that $Par(n)$ has no left child, in which case we can let $A_L = 0$, $I_L = 0$, and $Mr_L = \infty$, and Theorem 1 still holds.

Theorem 2: *Given two solutions* S_1, *and* S_2, *in the root's solution set, where* $S_1 \succ S_2$, *then either* S_2 *is an infeasible solutions or* S_2 *is a suboptimal solution.*

Proof: From (16) we see that if S_1 *is infeasible, then* S_2 *is also infeasible. If* S_1 *is feasible, then* S_2 *is either feasible or infeasible. If* S_2 *is feasible, then from (14) it is suboptimal, that is, there is at least one other solution with a smaller area.*

Incremental dominance prune

The dominance prune strategy is a variation on determining the maxima of a set of points. This problem can be solved in $\Theta(n \log n)$ time with an elegant divide and conquer algorithm described in [13]. This algorithm requires that the entire unpruned set of partial solutions be generated and stored in memory. Unfortunately, given the size of some of the unpruned sets, this would require an enormous amount of memory. To combat this problem we developed an incremental pruning algorithm using the maxima algorithm and taking advantage of the observation that the pruned sets are always much smaller than the unpruned sets.

Our strategy is to generate K elements of the unpruned set and use the maxima algorithm to prune this set. Once the maxima algorithm produces a pruned set, containing K_1 elements, where $K_1 < K$, we generate another K elements from the unpruned set and append them to the set of K_1 elements. This new set of $K + K_1$ elements is then pruned using the maxima algorithm. This process of generating elements and pruning sets is repeated $\lceil \frac{m}{K} \rceil$ times, where m is the number of elements in the unpruned set. Fig 5 lists the pseudo code for this algorithm.

```
S ← ∅
i ← 0
While more elements to generate
begin
    S ← S ∪ generated element
    i ← i + 1
    If i = K then
    begin
        S ← maxima(S)
        i ← 0
    end
end
S ← maxima(S)
```

Figure 5: Incremental maxima algorithm pseudo code

The time complexity of this algorithm is seen to be

$$\begin{aligned} & O \quad (\lceil \frac{m}{K} \rceil (K + K_1) \log(K + K_1)) \\ \in \; & O \quad (\lfloor \frac{m - 1 + K}{K} \rfloor (K + K_1) \log(K + K_1)) \\ \in \; & O \quad (\frac{(m - 1 + K)(K + K_1)}{K} \log(K + K_1)) \end{aligned}$$

but since $K_1 \leq K$, we have

$$O((m - 1 + K) \log(2K))$$

and since $K \leq m$, we have

$$O(m \log K)$$

Fast sorting

The maxima algorithm outlined in [13] requires that the unpruned set be initially sorted on one of the components, A, I, or Mr, before the divide and conquer step can be applied. In general this sort takes $\Theta(m \log m)$ time, however, if we choose to sort on the area component, we can reduce the time complexity of the sort. Recalling that the areas must be integer multiples of the grid spacing, and that there are at most $2l_{net}S_{max}$ possible area values, where l_{net} and S_{max} are both integers, we can sort on the area component using a bucket sort with $2l_{net}S_{max}$ buckets. This gives us a sort with time complexity $O(m + 2l_{net}S_{max})$.

While this sorting technique does not reduce the overall time complexity of the maxima algorithm, since the divide and conquer step still requires $O(m \log m)$ time, it significantly reduces the execution time on large sets.

5.3 Resistance-to-root prune

The "resistance-to-root prune" is based on the fact that we can easily calculate the resistance, R_{root}, between any node in the binary network and the root node. In particular, given a partial solution, whose current is I and remaining noise margin is Mr, if

$$Mr - I\,R_{root} < 0$$

then we know that this partial solution will only generate infeasible solutions at the root and therefor we can prune this partial solution.

5.4 Maximum-area prune

The "maximum-area prune" works by asking the question "what is the smallest spacing, S, for which there is a feasible solution to (8) and (9) when we set all $S_{ij} = S$?" If $S = 1$ then we have found the solution to (8) and (9). If $S > S_{max}$ then we know that there is no solution to (8) and (9). If $1 < S \leq S_{max}$ then we can compute an upper bound, $2L_{net}S$, on the area of any partial solution, and prune accordingly. Note that the value of S can be computed in $O(S_{max}l_{net})$ time since (8) and (9) can be solved in $O(l_{net})$ time when all S_{ij} are set to a constant.

6 Computational complexity

An upper bound on the computational complexity to approximate a solution to (8) and (9) can be found as follows:

We know, from (13), the the maximum number of elements in any partial solution set is

$$O(S_{max}l_{net}I_{max}N_A)$$

The time to build an unpruned partial solution set is

$$O({S_{max}}^3{l_{net}}^2{I_{max}}^2{N_A}^2)$$

The time to build a pruned partial solution set is

$$O({S_{max}}^3{l_{net}}^2{I_{max}}^2{N_A}^2\log({S_{max}}^3{l_{net}}^2{I_{max}}^2{N_A}^2))$$

Since there can be at most l_{net} such sets, the total computation time is

$$O({S_{max}}^3{l_{net}}^3{I_{max}}^2{N_A}^2\log({S_{max}}^3{l_{net}}^2{I_{max}}^2{N_A}^2))$$

Since S_{max} and I_{max} are both constants, we have

$$O({l_{net}}^3{N_A}^2\log({l_{net}}^2{N_A}^2)) \tag{20}$$

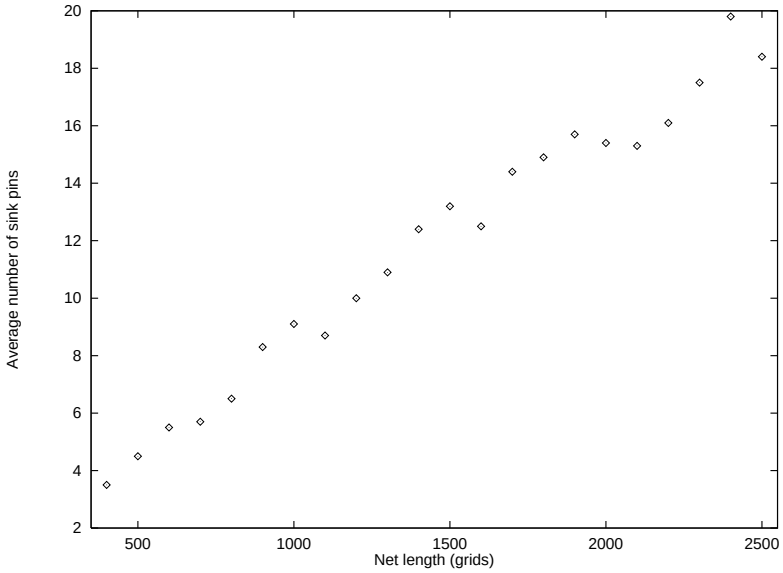

Figure 6: Average number of sink pins.

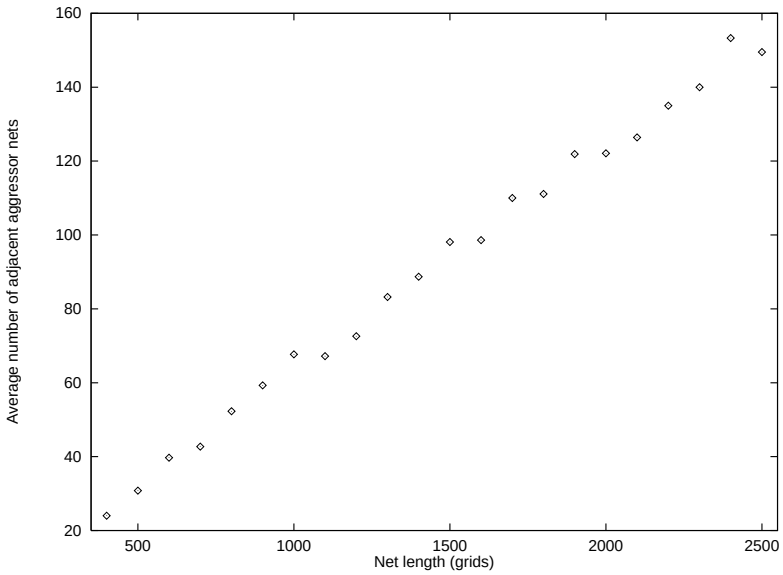

Figure 7: Average number of adjacent aggressor nets.

7 Results

We implemented our algorithm in C++ on an IBM RS/6000 model 590 with 500 MB of memory, running under AIX. The parameters used in our experiments are based on the .18 μm technology specified in the SIA road map [14]. Specifically, the grid spacing was 0.55 μm. The resistance of the wiring was 0.16 $\Omega/grid$. The capacitive coupling between adjacent wires separated by one grid spacing was 0.41 $fF/grid$. The supply voltage was 1.5V. The noise margins for the sink pins was 0.5V. The aggressor net rise times were randomly selected between 1 pS and 100 pS. The driver resistance was 100 Ω. For these experiments we chose $S_{max} = 5$, and $I_{max} = 100$.

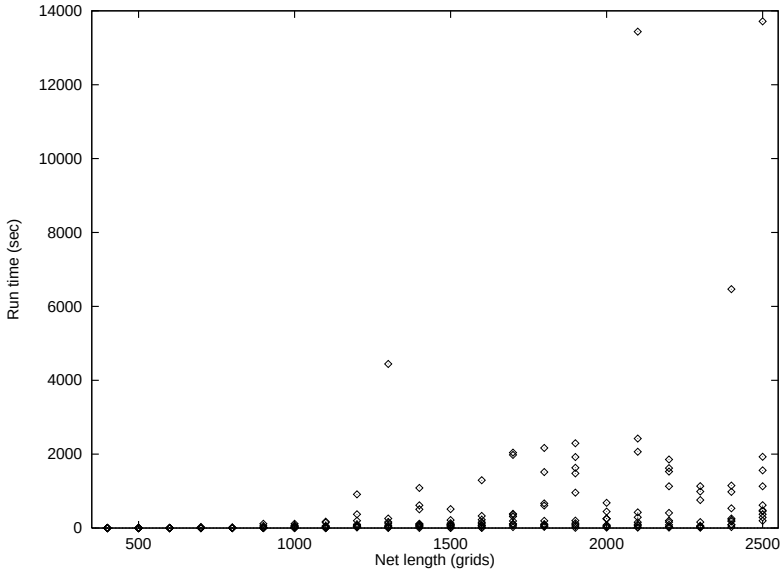

Figure 8: Run time to compute spacing sets.

We tested the algorithm on 220, nontrivial, randomly generated nets. By "nontrivial" we mean that each net required at least on spacing, in the set of spacings forming the solution, to be greater than 1, and that noise constrains in (9) could be satisfied when all $S_{ij} = S_{max}$. The length of these nets ranged from 400 grid spacings to 2500 grid spacings, or from about $0.22\,mm$ to about $1.37\,mm$. These nets can be further characterized by the graphs in Fig. 6 and Fig. 7 which show the average number of sink pins for a victim net of given length, and the average number of adjacent aggressor nets for a victim net of given length, respectively.

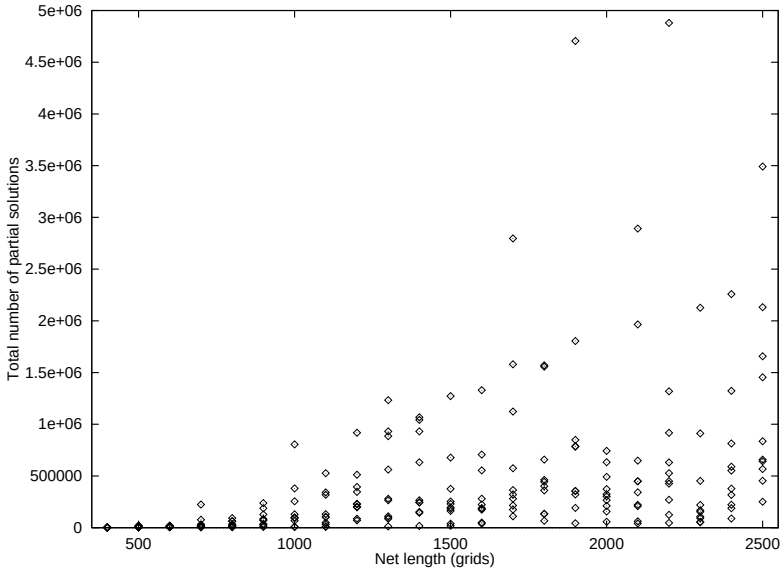

Figure 9: Total number of partial solutions needed to compute spacing sets.

Fig. 8 shows the run time results from the experiments. From this we see that an optimal set of spacing could be computed in under $2500\,sec$ for all but four of the test nets. Given

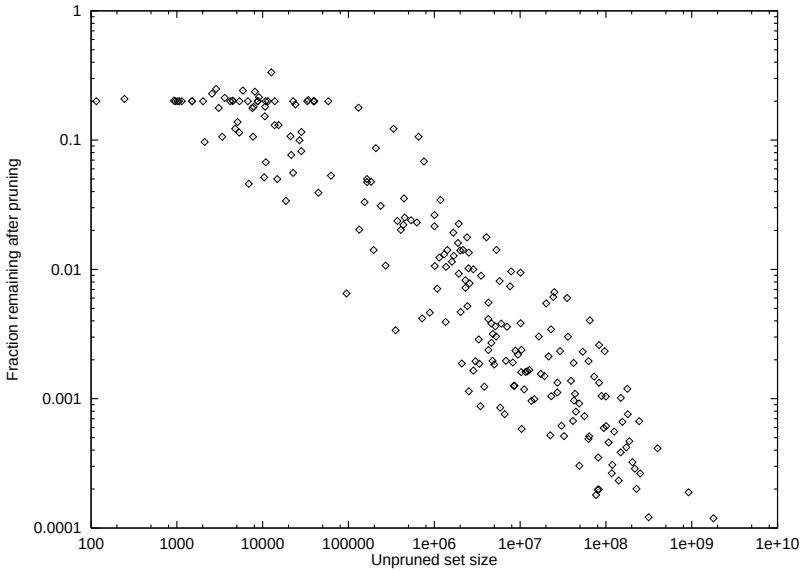

Figure 10: Fraction of unpruned set remaining after pruning.

the heuristic nature of the pruning strategies, it is no surprise that a few of the nets were not effectively pruned. From this figure it is also interesting to note that the run times peak for a net length of about 2000 grids. This can be attributed to the fact that the maximum-area pruning strategy is effective on shorter nets and the resistance-to-root pruning strategy becomes effective for longer nets.

Fig. 9 shows that the total number of partial solutions (after pruning) that need to be stored for a given net length. From this we see that the vast majority of nets needed to store less than 2.5 million partial solution, while a few nets required up to 5 million partial solutions be stored.

Finally, from Fig. 10 we can see the justification for the underlying assumption for our incremental pruning strategy. In particular, we see that for large sets, sets with 1 million or more elements, we can easily get pruned sets that are $\frac{1}{10}$ to $\frac{1}{10000}$ the size of their unpruned counterparts.

8 Conclusion

In this paper we have formulated the optimal wire spacing under cross talk noise constraints problem and presented a polynomial time dynamic programming solution to the discrete spacing version of this problem. We have demonstrated the feasibility of this algorithm by implementing it and running it on what we believe to be both realistic and challenging experimental data.

References

[1] T. Gao and C. Liu, "Minimum Crosstalk Channel Routing," *IEEE Trans. Computer-Aided Design*, vol. 15, no. 5, pp. 465–474, 1996.

[2] T. Gao and C. Liu, "Minimum Crosstalk Switchbox Routing," in *Proc. IEEE/ACM Int. Conf. on Computer Aided Design*, pp. 610–615, 1994.

[3] H. Chen and C. Wong, "Wiring and Crosstalk Avoidance in Multi-Chip Module Design," in *Proceedings of the IEEE 1992 Custom Integrated Circuits Conference*, pp. 28.6.1–28.6.4, 1992.

[4] G. Devaraj and D. Bhatia, "Crosstalk Driven MCM Router," *Journal of Microelectronic Systems Integration*, vol. 2, no. 2, pp. 65–80, 1994.

[5] T. Miyoshi, S. Wakabayashi, T. Koide, and N. Yoshida, "An mcm Routing Algorithm Considering Crosstalk," in *IEEE Symposium on Circuits and Systems*, pp. 211–214, 1995.

[6] K. Jhang, S. HA, and C. Jhon, "A Segment Rearrangement Approach to Channel Routing Under the Crosstalk Constraints," in *IEEE Asia-Pacific Conf. onf Circuits and Systems*, pp. 536–541, 1994.

[7] J. Cong, L. He, C. Koh, and Z. Pan, "Global Iinterconnect Sizing and Spacing with Consideration of Coupling Capacitance," in *Proc. IEEE/ACM Int. Conf. on Computer Aided Design*, pp. 628–633, 1997.

[8] K. Chaudhary, A. Onozawa, and E. Kuh, "A Spacing Algorithm for Performance Enhancement and Cross-talk Reducton," in *Proc. IEEE/ACM Int. Conf. on Computer Aided Design*, pp. 697–702, 1993.

[9] T. Okamoto and J. Cong, "Buffered Steiner Tree Construction with Wire Sizing for Interconnect Layout Optimization," in *Proc. IEEE/ACM Int. Conf. on Computer Aided Design*, pp. 44–49, 1996.

[10] J. Lillis, C. Chen, and T. Lin, "Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model," in *Proc. IEEE/ACM Int. Conf. on Computer Aided Design*, pp. 138–143, 1995.

[11] D. J. Staepelaere, J. Jue, T. Dayan, and W. W.-M. Dai, "Surf: a rubber-band routing system for multichip modules," in *Proc. IEEE Design and Test of Computers*, 1993.

[12] A. Devgan, "Efficient coupled noise estimation for on-chip interconnects," in *Proc. IEEE/ACM Int. Conf. on Computer Aided Design*, pp. 147–151, 1997.

[13] F. Preparata and M. Shamos, *Computational Geometry.* New York, NY: Springer-Verlag, 1985.

[14] Semiconductor Industry Association, *National Technology Roadmap for Semiconductors.* 1994.