A Pareto-Algebraic Framework for Signal Power Optimization in Global Routing

Hamid Shojaei, Tai-Hsuan Wu, Azadeh Davoodi
Department of Electrical and Computer Engineering, University of Wisconsin - Madison
{shojaei,twu3,adavoodi} @wisc.edu

Twan Basten
Department of Electrical Engineering
Eindhoven University of Technology
& Embedded Systems Institute
Eindhoven, The Netherlands
a.a.basten@tue.nl

ABSTRACT

This paper proposes a framework for (signal) interconnect power optimization at the global routing stage. In a typical design flow, the primary objective of global routing is minimization of wirelength and via consumption. Our framework takes a global routing solution that is optimized for this objective, and quickly generates a new solution that is optimized for signal power, with only a small, controlled degradation in wirelength. Our model of signal power includes layer-dependent fringe and area capacitances of the routes, and their spacing. Our framework is fast compared to the existing global routing procedures, thereby not causing much overhead and fitting well in the design flow to optimize signal power after wirelength minimization. The framework is based on Pareto-algebraic operations and generates multiple global routing solutions to provide a tradeoff between power and wirelength, thereby allowing the user to optimize power with a controlled degradation in wirelength. The generated solution remains free of overflow in routing resource usage. We experiment with large benchmarks from the ISPD 2008 suite and a 45nm technology model. We show on average 19.9% power saving with at most 3% wirelength degradation using the existing wirelength optimized solutions from the open literature.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids

General Terms

Algorithms, Design

Keywords

Global Routing, Dynamic Power, Pareto Algebra

1. INTRODUCTION

Continuous technology scaling has resulted in the interconnects to highly impact a multitude of aspects such as timing, power and manufacturability of a design. Consequently, global routing (GR), the initial stage where the interconnects are planned, has increasingly gained importance. In a typical design flow, GR is initially conducted for the primary objective of minimizing wirelength and via usage. Optimizing for other objectives can follow in subsequent calls of GR based on this initially-generated solution.

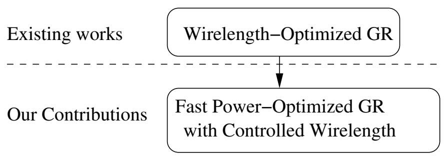


Figure 1: Relationship of our GR framework with traditional wirelength-optimized procedures.

Power of signal interconnects (which excludes other types of interconnects such as power-delivery and clocking networks) can have a significant contribution to the total power consumption; e.g., about 20% of the total power in microprocessors is signal power [6]. To model the signal power accurately, area and fringe capacitances for each wire segment should be considered depending on its width and metal layer. Furthermore, the spacing between the wires can determine their coupling capacitance which is an important contributor to the power and highly deteriorates with technology scaling. The above factors (i.e. metal layer and wire width) are determined during the GR stage. Moreover, wire spacing can also be approximated at this stage. Therefore, opportunities lie in signal power optimization at this stage.

With the release of industrial benchmarks in the ISPD contest [1], many new GR procedures were developed [4, 8, 3, 11, 10] to handle realistic and large-sized instances. The focus of the new procedures was to address the primary objective of minimizing wirelength and via usage, preferably without generating any overflow of routing resources. No recent work has focused on power optimization during GR on these larger instances with power models to include wire width, layer, and spacing.

In this work, we first propose a signal power model at the GR stage which estimates cross-coupling capacitance based on the routing grid edge utilization. We then propose a fast GR procedure to optimize the signal power with a controlled level of wirelength degradation while handling large-sized instances. As shown in Fig. 1, our framework can receive a wirelength-optimized routing solution (from any wirelength-centric procedures such as [4, 8, 3, 11, 10]) and further optimize it for power. In a more general sense, it can receive multiple routing solutions which are for example generated by running different wirelength-centric tools in parallel, or by simultaneously running the same tool in different modes.

Our framework then saves signal power by solely rerouting the nets based on their initial routes while ensuring the feasibility of the initial routing solutions (in terms of no overflow generation) remains intact. For power saving opportunities, routes in highly congested areas can be detoured through less-congested ones to allow more wire spacing and hence lower coupling capacitance. As another example, routes can be detoured from higher metal layers to the lower ones to decrease their area and fringe capacitances.

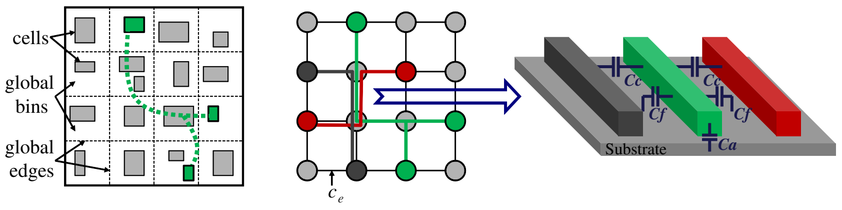


Figure 2: The construction of grid graph for GR and modeling different interconnect capacitance components.

Our framework is based on an extension of the Pareto-algebraic heuristic of [9] to solve the multi-dimensional multiple-choice knapsack problem. It takes one or more candidate routes for each net as a set of its initial *configurations*. It can also dynamically generate additional routes for each net at the time it is processed. The procedure visits the nets sequentially and combines their configurations. It only keeps the feasible configurations which do not cause overflow and can also provide a tradeoff between wirelength and power. The tradeoff allows the user to specify a tolerable threshold of wirelength degradation as a constraint for power optimization. In the end, one configuration for each net is selected. The Pareto-algebraic nature of our heuristic makes its execution runtime fast. More specifically, our contributions can be summarized as follows:

- We represent a candidate route of a net as a configuration in multiple dimensions with information about its wirelength, via count, power, and routing grid usage. Our procedure handles two classes of configurations: those which are provided a-priori, and those which are generated on the fly.
- We introduce a congestion-based ordering for processing the nets for evaluating their configurations within the framework. The ordering allows more effective combination of the configurations for faster runtime and better solution quality.
- We propose a procedure for dynamic generation of power-optimized candidate routes as the configurations of a net, from its static wirelength-centric configurations. This generation is based on the status of the combined configurations, and a dynamically-updated measure of routing congestion.
- To handle large problem sizes for runtime and memory usage, our procedure offers a reduction technique to identify and store selected Pareto-optimal configurations up to a maximum number of configurations. This allows us to bound both the execution time and memory requirements of the computations beforehand.

In our simulations, we experiment with ISPD 2008 benchmarks and a 45nm technology library model. We consider two cases of configurations generation based on the solutions of three state of the art academic GRs, and based on our dynamic configuration generation. Overall we show an average of 19.9% signal power saving with only 2.7% degradation in wirelength, without any overflow.

2. POWER MODELING FOR GR

The GR problem can be conceptualized on a grid-graph $G = (\mathcal{V}, \mathcal{E})$ of global bins, each containing some cells, as depicted in Fig. 2. The boundary between two adjacent global bins in $\mathcal{V}$ is modeled as an edge $e \in \mathcal{E}$. With each edge e, a capacity c_e is associated, reflecting the maximum available routing resource between two adjacent bins. A set of (multi-terminal) nets $\mathcal{N}$ is also provided as input. Each net n_i is defined by a set of vertices in $\mathcal{V}$, corresponding to a set of cells in the global bins as its terminals. The goal of the GR problem is to find a Steiner tree t_i in G connecting the terminals of net $n_i, \forall i \in \mathcal{N}$.

The primary objective of GR is minimizing wirelength. The wirelength is determined by counting the number of edges on the Steiner trees of all the nets $\mathcal{N}$. The actual length of each edge can be computed from the chip dimension and the granularity of the routing grid. Another important metric in GR is overflow which is defined as the total amount by which routed demand exceeds the edge capacities. It is desirable to also have a zero overflow solution.

In this work, we adopt the framework of Fig. 1 to also optimize signal power, assuming an initial wirelength-centric GR solution is provided. The total signal interconnect power denoted by P can be expressed as summation of dynamic powers of individual nets:

$$P = V_{DD}^2 \times f_{clk} \times \sum_{i=1}^{|\mathcal{N}|} \alpha_i (C_i^{sink} + C_i^{wire}) \qquad (1)$$

where V_{DD} is the supply voltage, f_{clk} is the frequency, and α_i is the switching activity of net i. The capacitance of a routed net i is the summation of the capacitances of its sink cells (denoted by C_i^{sink}), and of its wire (denoted by C_i^{wire}). The C_i^{wire} is expressed as the summation of all its unit wire segment capacitances and given by the equation below:

$$C_i^{wire} = \sum_{\forall j, e_j \in t_i} C_j^{unit} \qquad (2)$$

where C_j^{unit} is the capacitance corresponding to edge e_j which is contained in tree t_i. For a wire segment corresponding to edge e_j, we assume a width w and metal layer l. In the general case, the wire can be in the middle of two other neighboring segments running in parallel with spacing s, all of which are mapped to edge e_j, as shown in Fig. 2. The capacitance corresponding to this wire segment is denoted by C_j^{unit} and is given by the equation below:

$$C_j^{unit} = Ca_j(l, w) + 2Cf_j(l, w, s) + 2Cc_j(l, w, s) \qquad (3)$$

where Ca_j and Cf_j are the area and fringe capacitances with respect to substrate, and Cc_j is the cross-coupling capacitance between the wire segment and its parallel-running wires. Figure 2 illustrates these three types of capacitances.

The area, fringe and coupling capacitances are a function of given metal layer l and wire width w. In addition, the fringe and coupling capacitance also depend on spacing s. For a given technology library, metal layer, wire width and spacing, the corresponding unit-length capacitance is typically provided as a lookup table.

For example, for the 45nm technology model used in this work [2], we observed that with decrease in spacing, Cc significantly increases while Cf slightly decreases. Also, with increase in wire width, Ca drastically increases. The wire width is determined from the metal layer and a higher metal layer requires a higher wire width. The spacing depends on the routing congestion.

For a given congestion determined from a GR solution, we determine the spacing for each edge e based on its utilization and the edge dimension which is determined from the chip dimension and the routing grid granularity. More specifically, if edge e is utilized to its capacity c_e, it means c_e wire segments of different nets pass edge e (see Fig. 2). We determine spacing s on edge e such that the maximum distance between its assigned wires are obtained.

This strategy minimizes the fringe and cross-coupling capacitance which makes sense if no spacing constraints are provided. On the other hand, if additional spacing constraints are provided, they can be incorporated for a more accurate estimate of wire spacing per grid edge. As part of the contributions of this work, we show how spacing can be approximated based on a dynamically-updated model of routing congestion, during global routing.

3. PROBLEM FORMULATION

We assume a set of candidate Steiner trees for each net is available. These candidate trees may be provided a-priori based on a set of wirelength-optimized trees obtained from a flow as given in Fig. 1. In addition, the candidate trees can also be generated on the fly within our procedure when a net is getting processed. (Details on generating candidate route are discussed in Section 5.2.)

Let us denote by $\mathcal{T}(n_i)$ the set of candidate trees for connecting the terminals in net n_i. Let $a_{te} = 1$ if tree t contains edge $e \in E$ in the grid-graph, $a_{te} = 0$ otherwise. Define the binary variable x_{it} that is equal to 1 if and only if tree $t \in \mathcal{T}(n_i)$ is selected for net n_i. The power-centric global routing problem is expressed as:

$$\min_x \sum_{i=1}^{|\mathcal{N}|} \sum_{t \in \mathcal{T}(n_i)} p_{it} x_{it} \qquad \text{(PGR)}$$

$$\begin{cases} \sum_{t \in \mathcal{T}(T_i)} x_{it} = 1 & \forall i = 1, \ldots, |\mathcal{N}| \\ \sum_{i=1}^{|\mathcal{N}|} \sum_{t \in \mathcal{T}(n_i)} a_{te} x_{it} \leq c_e & \forall e \in E \\ \sum_{i=1}^{|\mathcal{N}|} \sum_{t \in \mathcal{T}(n_i)} wl_{it} x_{it} \leq WL_{th} & \forall i = 1, \ldots, |\mathcal{N}| \\ x_{it} = \{0, 1\} & \forall i = 1, \ldots, |\mathcal{N}|, \forall t \in \mathcal{T}(T_i). \end{cases}$$

where p_{it} and wl_{it} are the power and wirelength (including via count) of tree t for net n_i. For a given tree t, the knowledge of its wiring segments on the routing grid over different metal layers allows computing wl_{it} as well as the area and fringe capacitances which are required to compute p_{it}, as explained in Section 2. In addition, to compute p_{it}, we also need the spacing of t with respect to neighboring routes. We discuss calculation of the spacing later, when we explain the specifics of our GR procedure in Section 5.2.

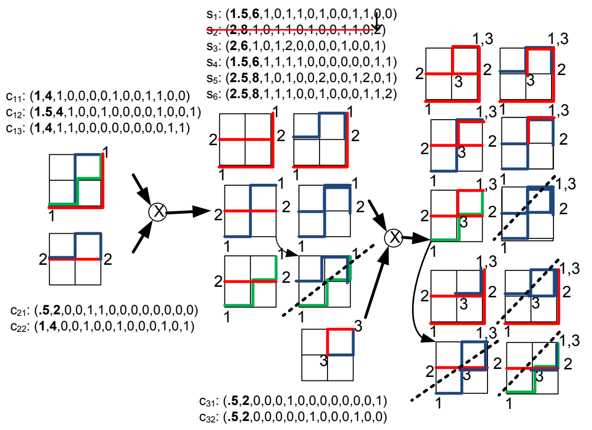
The objective of the PGR formulation is to minimize the total power of the selected trees. The first set of equations enforces selection of one tree for each net. The second set of equations ensures that the selected trees of all the nets do not result in exceeding the edge capacities c_e. The third equation bounds the total wirelength of selected trees by a user defined input WL_{th}. Thereby, power minimization can be done with respect to a *tolerable degradation factor* compared to an initial wirelength-centric solution.

In this paper, we assume the set of candidate trees of the nets allows a feasible solution for the PGR formulation. This is possible if the initial wirelength-centric solutions are free of overflow as they will be included in the set of candidate trees. Indeed, the existing wirelength-centric procedures have shown to generate overflow-free solutions for the majority of the ISPD benchmarks [1]. However, in case of overflow, our PGR procedure can be extended to minimize overflow as another dimension.

The PGR formulation is in the form of the multi-dimensional multiple-choice knapsack problem (MMKP) [7] which is in the class of NP-complete problems. In [9], a Pareto-algebraic heuristic is given to quickly solve MMKP instances with high quality. We extend the procedure in [9] to generate a set of GR solutions trading off power and wirelength, while imposing the edge capacity and wirelength constraints. We pick the solution with minimum power.

4. PARETO-ALGEBRAIC PROCEDURE

In a Pareto-algebraic description of (PGR), each candidate tree of a net is represented in terms of a configuration of $R+2$ dimensions $(d_1, d_2, \ldots, d_{R+2})$, where R is the number of edges in the routing grid-graph.



s1: (1.5,6,1,0,1,1,0,1,0,0,1,1,0,0)
~~s2: (2,8,1,0,1,1,0,1,0,0,1,1,0,2)~~
s3: (2,6,1,0,1,2,0,0,0,0,1,0,0,1)
s4: (1.5,6,1,1,1,1,0,0,0,0,0,0,1,1)
s5: (2.5,8,1,0,1,0,0,2,0,0,1,2,0,1)
s6: (2.5,8,1,1,1,0,0,1,0,0,0,1,1,2)
c11: (1,4,1,0,0,0,0,1,0,0,1,1,0,0)
c12: (1.5,4,1,0,0,1,0,0,0,0,1,0,0,1)
c13: (1,4,1,1,0,0,0,0,0,0,0,0,1,1)
c21: (.5,2,0,0,1,1,0,0,0,0,0,0,0,0)
c22: (1,4,0,0,1,0,0,1,0,0,0,1,0,1)
c31: (.5,2,0,0,0,1,0,0,0,0,0,0,0,1)
c32: (.5,2,0,0,0,0,0,1,0,0,0,1,0,0)
X
X
1
2
3
1,3

Figure 3: Example of the Pareto-algebraic procedure.

The first dimension is the power corresponding to that tree, the second dimension is its wirelength (including vias), and the remaining R dimensions reflect the grid edge utilization. We denote the j^{th} configuration of net n_i by $c_{i,j}$. For example, in Figure 3 (left), we have 3 configurations for n_1 and 2 for n_2. The two bold entries for each configuration correspond to the power and wirelength, respectively. The remaining entries have a 1 in a dimension if the corresponding edge is utilized by that tree.

The set of candidate trees for a net n_i thus composes its configuration set which we denote by $\mathcal{C}_i = \{c_{i,1}, ..., c_{i,|\mathcal{C}_i|}\}$. Configuration $c_{i,j}$ is a Pareto point of set $\mathcal{C}_i$ iff it is not dominated in terms of power and wirelength and edge utilizations by any other configuration in $\mathcal{C}_i$. Pareto algebra defines operations on configuration sets in order to compute or reason about Pareto points [5].

We first discuss an overview of our procedure. We traverse the nets once in a specific order (given in Section 5.1). At each step of computation, we combine the configurations of the current net with an existing configuration set corresponding to all the nets processed in the previous steps. During each step we maintain a set of options for all nets processed up-to that point, each option providing a different power-wirelength tradeoff. After each step, we thus obtain a set of compound configurations denoted by $\mathcal{S} = \{s_1, s_2, ..., s_{|\mathcal{S}|}\}$ where s_i is one compound configuration. Conceptually, each combination of two configuration sets corresponds to the Cartesian product of these sets. Each compound configuration is represented by a tuple corresponding to the element-wise addition of all tuples corresponding to the constituent net configurations. Infeasible compound configurations violating the edge and wirelength capacity constraints are removed. Configurations that are dominated in terms of power and wirelength and edge utilization are also removed, i.e., only Pareto points are kept.

Fig. 3 shows a small example of nets 1, 2, 3 on a 2x2 grid. In the first step, net 1 (with 3 configurations) and net 2 (with 2 configurations) are combined. Among the 6 compound configurations, s_2 is dominated by s_1 and removed. Net 3 with 2 configurations is then combined with the configuration set representing nets 1 and 2. The last step then removes 3 configurations from the 10 resulting ones (where 1 compound configuration is dominated and 2 others are infeasible due to violating the presumed edge capacity of 2).

Algorithm 1 gives the details of our procedure, using Pareto-algebraic concepts and tailoring the technique of [9] to solve the PGR formulation. For each net, we require an initial configuration set $\mathcal{C}_i = \{c_{i,1}, \ldots, c_{i,m_{init}} \mid 1 \leq i \leq |\mathcal{N}|\}$. These are wirelength-optimized configurations. One initial configuration suffices, but multiple solutions could for example be obtained by running multiple routing tools (possibly in parallel), or by (simultaneously) running the same tool in different modes.

Algorithm 1 Pareto-algebraic heuristic for PGR

```
1:  PGR($\mathcal{C}_i$ ($\forall n_i \in \mathcal{N}$), $c_\mathcal{E}, \alpha,\ L,\ WL_{th}$)
2:  $s_1 = \mathcal{C}_1 = \{c_{i,1}, ..., c_{i,m_{init}}\}; \mathcal{S} = \{s_1\}$
3:  for all net $n_i$ $i = 2$ to $|\mathcal{N}|$ do
4:      $\mathcal{S}^{tmp} = \emptyset$
5:      $m_{max} = |\mathcal{C}_i| = m_{init}$
6:      for all $s_j \in \mathcal{S}$ do
7:          for all $l = 1$ to $m_{init}$ do
8:              $m_{max}$++
9:              $c_{i,m_{max}}$ = dynamic-config-generate($\alpha,\ c_{i,l}, s_j, c_{k(>i),1}$)
10:         end for
11:         for all $l = 1$ to $m_{max}$ do
12:             if constrain($s_j, c_{i,l}, c_{k(>i),1}, c_\mathcal{E}, WL_{th}$) then
13:                 $\mathcal{S}^{tmp}$ = sum($s_j, c_{i,l}$) $\cup\ \mathcal{S}^{tmp}$
14:                 min($\mathcal{S}^{tmp}$)
15:             end if
16:         end for
17:     end for
18:     $\mathcal{S} = \mathcal{S}^{tmp}$
19:     if $|\mathcal{S}| > L$ then
20:         reduce($\mathcal{S}$) // Sec. 5.3
21:     end if
22: end for
23: Post-processing();
```

In addition, we take as input the edge capacity and activity vectors $c_\mathcal{E}$ and α for the edges and nets, respectively, and two user-defined parameters L and WL_{th}. The former controls the running time of the algorithm (as explained below) while the latter defines the wirelength threshold. The set of compound configurations $\mathcal{S}$ is initially set to $\mathcal{C}_1$ for net n_1 (line 2). The remaining nets are then traversed once. For each net n_i, we consider each $s_j \in \mathcal{S}$ reflecting the combinations of processed nets n_1 to n_{i-1} with the configurations of n_i. Before the combination process, we dynamically generate additional configurations for n_i to provide a tradeoff between the wirelength and power options of n_i (lines 7 to 9). More specifically, for each s_j we add one new configuration for each of the initial configurations $c_{i,1}$ to $c_{i,m_{init}}$. Parameter m_{max} reflects the number of configurations of n_i during dynamic tree generation.

For dynamic tree generation we rely on the specifics of the routing state imposed by compound configuration s_j (for n_1 to n_{i-1}) and configuration $c_{k(>i),1}$ (for remaining nets $k > i$). Note, since each net $n_{k>i}$ is not yet routed, we estimate its configuration as $c_{k,1}$. We assume $c_{k,1}$ is a wirelength-optimized configuration. We note our final solution is evaluated *without* any estimation. Dynamic tree generation also uses the vector of activity factors α.

Next, we traverse all the m_{max} configurations of net n_i. We combine each configuration $c_{i,l}$ with s_j (lines 11 to 15). Within the combination process, we first check for feasibility in terms of satisfaction of the edge capacities and the wirelength constraint (line 12). For the capacity constraint, we check the provided edge capacities $c_\mathcal{E}$ against a utilization, obtained from s_j (for n_1 to n_{i-1}), $c_{i,l}$ (for n_i), and $c_{k(>i),1}$ (configuration estimates for all $n_{k>i}$). Specifically, the corresponding edge elements are added in the considered configurations. Feasibility is then checked for each element. For the wirelength constraint, we compare the summation of wirelength elements in s_j and $c_{i,l}$ against WL_{th}.

If the result is feasible, the combination process continues by adding the power and wirelength entries of s_j and $c_{i,l}$ via a *sum* operation. The result is added to the set $\mathcal{S}^{tmp}$ reflecting the state of combinations between net n_i configurations and the s_js. Next, the Pareto-algebraic *min* operation is applied to $\mathcal{S}^{tmp}$ to extract the Pareto points.

Algorithm 2 Net ordering procedure

```
1: NETORDERING($\mathcal{C} = \{c_{i,1} \mid n_i \in \mathcal{N}\},\ \mathcal{N},\ \mathcal{E}$)
2: $\mathcal{N}_{order} = \emptyset$; $\mathcal{E}_{candid} = \mathcal{E}$
3: while $|\mathcal{N}_{order}| \neq |\mathcal{N}|$ do
4:     $e_{crit}$=arg max$_{\forall e \in \mathcal{E}_{candid}}$($u_e$ computed based on $\mathcal{C}$)
5:     $\mathcal{N}_{crit}$= $\{n_i \in \mathcal{N} |\ e_{crit}$ included in $c_{i,1}\}$.
6:     $\mathcal{N}_{order} = \mathcal{N}_{order} \circ$ sort-decreasing-wirelength($\mathcal{N}_{crit}$)
7:     $\mathcal{E}_{candid} = \mathcal{E}_{candid} - e_{crit}$
8: end while
```

The process repeats to combine the next configuration s_{j+1} with n_i. When all s_js are combined with the configurations of n_i, set $\mathcal{S}$ is updated to $\mathcal{S}^{tmp}$ and net n_{i+1} is processed.

The above procedure is intractable, as the size of $\mathcal{S}$ can become very large. To speed up the computation, we check if the size of $\mathcal{S}$ becomes larger than the user-defined threshold parameter L. We prune some of the configurations and keep only L configurations (lines 19 to 21). We tailor the reduction heuristic of [9] for our problem to identify the L configurations which can provide a good tradeoff between power and wirelength. We explain this step in Section 5.3. We then pick the minimum power solution to obtain one route for each net.

In the end, we apply a post-processing step in which we traverse the nets once again (See section 5.1 for the traverse order). We consider the candidate routes stored for each net. If a net has a route that can further improve the power and it does not cause overflow of routing resources and exceed the wirelength threshold, we update the route of the net.

5. MORE DETAILS ON OUR PROCEDURE

5.1 Net Ordering

Our Pareto-algebraic procedure relies on an ordering of the nets for their processing. Our experience for GR is that the ordering can highly impact the quality of the final solution. A good ordering is one which can identify and prune the infeasible and dominated solutions early in the combination steps. Otherwise, the number of Pareto points will quickly reach the threshold and the reduction step needs to be often applied to remove some of the Pareto points.

A good ordering can be achieved by looking at congested areas first. The routing grid edges are utilized close to their capacities in these areas, and it will be more likely to detect and remove infeasible and dominated configurations. In addition, the congested areas include many more nets and thereby contribute more to overall power.

Algorithm 2 gives the pseudo-code of our ordering procedure. We start from the areas that are highly congested based on estimation of edge utilization using the first (wirelength-optimized) configuration for each net (i.e., $c_{i,1}$ for n_i). We keep updating the ordered subset of nets $\mathcal{N}_{order} \subseteq \mathcal{N}$ until $|\mathcal{N}_{order}| = |\mathcal{N}|$ (line 3). We first identify a critical edge $e_{crit} \in \mathcal{E}_{candid}$ with highest utilization u_e, where $\mathcal{E}_{candid} = \mathcal{E}$ at the beginning (line 2). The utilization u_e is determined by counting the number of configurations $c_{i,1}$ which include e (line 4).

Next, we form the subset $\mathcal{N}_{crit} \subseteq \mathcal{N}$ of critical nets where $n_i \in \mathcal{N}_{crit}$ if $c_{i,1}$ includes e_{crit} (line 5). The configurations of these routes include e_{crit} and are more likely to result in pruning. We then sort the subset $\mathcal{N}_{crit}$ according to decreasing order of the wirelength of the nets, with wirelength approximated using $c_{i,1}$ for all $n_i \in \mathcal{N}_{crit}$ (line 6). The sorted subset is then concatenated to the end of the existing $\mathcal{N}_{order}$ (in line 6 where concatenation is denoted by the $\circ$ symbol). Finally, we update the set $\mathcal{E}_{candid}$ to find the next e_{crit} while excluding the previous e_{crit} identified from the previous iteration.

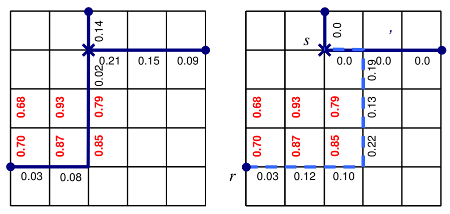


Figure 4: Generating power-optimized trees.

5.2 Dynamic Tree Generation

As stated in Algorithm 1, line 9, we are given a base set of configurations for all the nets which are s_j (for n_1 to n_{i-1}), $c_{i,l}$ (for n_i), and $c_{k(>i),1}$ (for remaining nets). We can therefore derive a Steiner tree t_l from the given configurations for each net n_l ($l = 1$ to $|\mathcal{N}|$). We are also given the activity factor vector α. We have a target net n_i with the base tree t_i obtained from its configuration $c_{i,l}$. Our goal is to generate a new route for n_i with less power.

We elaborate our procedure with the example of Fig. 4. On the left, we draw the tree t_i for target net i on the routing grid-graph. The power of tree t_i can be expressed as summation of power values of its edges. So we assign a weight to each edge on the grid reflecting the power of that edge and reroute t_i using this weighted graph for less power to obtain a better tree. We estimate the weight of edge e_j as follow:

$$w_j = \sum_{\forall n_l, e_j \in t_l} \alpha_l C_j^{unit} \tag{4}$$

where α_l is the switching activity of net n_l with t_l containing e_j, and C_j^{unit} is the unit capacitance corresponding to e_j. Recall from Eq. 3 that C_j^{unit} depends on the metal layer, size and spacing of the wire segments on edge e_j. The metal layer and size are known for e_j on the grid graph. We determine the spacing based on an estimate of the utilization of e_j by counting the number of trees t_l which includes e_j. For a utilization of u_{e_j}, we determine the spacing to allow maximum distance between the wires on e_j to obtain the smallest coupling capacitance. In defining the weights we disregard the contribution of supply, frequency and sink capacitances as they don't impact the outcome of power minimization.

To obtain a power-efficient tree, we reroute different branches of t_i on the weighted grid-graph. For a given tree, we first identify for each terminal a branch which connects it to the first Steiner point on the tree. For example, in Fig. 4, we identify the branch connecting terminal r to Steiner point s. For each branch, we calculate a power value as summation of its edge weights. We sort the branches according to their power values and reroute each branch for less power but more wirelength. This is done by applying a shortest-path algorithm on the weighted grid-graph between the terminal and the tree backbone (i.e., any point on the remainder of the tree). For example in Fig. 4, we show the rerouted branch with dashed lines which now passes from edges with lower weights.

5.3 The Reduce Procedure

If the size of compound configuration set $\mathcal{S}$ exceeds the user-defined parameter L, we apply a procedure to drop some of the Pareto-points. First, we project the compound configuration set into a 2-dimensional power versus wirelength space. Figure 5 shows a snapshot of the projected compound set $\mathcal{S}$ after processing 150K nets in benchmark `adaptec1`. Then the Pareto-algebraic *min* operation is performed to determine Pareto points (blue configurations in Fig. 5). If the number of Pareto points is still larger than L, we filter out L Pareto points from this set. We sort the configurations according to the weights power$_i$ $\times$ wirelength$_i$ and select L configurations by uniformly sampling their weights.

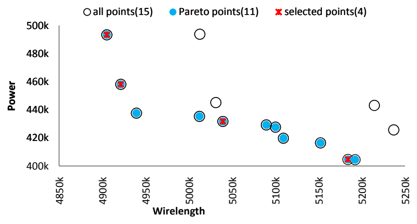


Figure 5: Reduce after processing 150K nets in adaptec1.

Table 1: Benchmark information

Benchmark	# Nets	Grid	# Layers	V.cap	H.cap
adaptec1	176715	324x324	6	70	70
adaptec2	207972	424x424	6	80	80
adaptec3	368494	774x779	6	62	62
adaptec4	401060	774x779	6	62	62
adaptec5	548073	465x468	6	110	110
newblue1	270713	399x399	6	62	62
newblue2	373790	557x463	6	110	110

This procedure allows a uniform sampling of the 2-dimensional wirelength-power space and provides the highest possible flexibility in selecting promising routing solutions. For more details, see [9], which justifies the procedure. The red Pareto configurations in Fig. 5 are the selected points for $L = 4$.

6. SIMULATION RESULTS

We implemented our Pareto-algebraic procedure using C++. We applied a subset of the ISPD 2008 benchmark suites [1] for which overflow-free solutions are available. Table 1 provides information about the benchmarks, including number of nets, grid granularity, number of metal layers, and vertical and horizontal edge capacities. We used the NANGATE 45nm technology open cell library [2]. It contains a lookup table for determining the area, fringe and coupling capacitance as a function of wire size and spacing as we explained in Section 2. We assumed (one) minimum wire size option given in the library for each metal layer. We randomly selected the activity factors of each net using a uniform distribution.

We considered the following 3 variants of our procedure:

1. **VER1:** We consider the wirelength-optimized solutions of 3 state of the art routing tools (NTHU-R 2.0 [3], FGR [8], and FastRoute 4.0 [11]) to provide 3 initial configurations for each net. The solutions were obtained from the authors and verified. Even though the total wirelength of these solutions does not significantly vary for the benchmarks, the 3 configuration trees for each net are very different. We skip dynamic tree generation in this variant of our algorithm.
2. **VER2:** We consider only one initial wirelength-optimized solution from NTHU-R [3] which had the smallest wirelength, and apply dynamic tree generation according to Algorithm 1. To bound the runtime, we only generate routes for the last 50K nets which are processed in each benchmark.
3. **VER3:** We combine the above variants by considering three initial configurations of the mentioned tools and apply dynamic tree generation. Again, to bound the runtime, we only generate routes for the last 50K nets which are processed.

In our procedure we set the tolerable wirelength degradation factor to 3% in reference to the wirelength of NTHU-R which is the smallest of the three considered tools for these benchmarks. We assume $L = 4$ due to memory restrictions, as all our experiments ran on machines with only 2GB memory.

Table 2 reports our results. First, for each of the considered routing tools, we report the wirelength (WL) and power (P) of their corresponding solutions. The wirelength is scaled to 10^5. Power is expressed in terms of switching capacitance (activity factor$\times$ capacitance). The unit of power is pF. We also report the runtimes of our procedures in minutes.

Table 2: Power, wirelength and runtime comparison.

Benchmark	NTHU-R[3]		FGR[8]		FastRoute[11]		VER1			VER2(50K)			VER3		
	P	WL	P	WL	P	WL	P(%)	WL(%)	TIME	P(%)	WL(%)	TIME	P(%)	WL(%)	TIME
Adaptec1	475.56	53.44	539.07	53.63	499.82	54.59	7.7	-0.5	2.1	5.1	-1.6	11.3	11.4	-2.9	16.8
Adaptec2	372.13	52.29	462.81	52.60	384.81	52.76	18.3	-1.1	5.7	13.6	-2.7	6.2	24.6	-3.0	12.8
Adaptec3	1085.62	131.01	1267.9	131.65	1093.27	132.13	11	-0.5	9.5	7.8	-0.9	22.9	13.5	-2.9	35.1
Adaptec4	799.37	121.69	1056.70	120.77	805.22	122.48	20.7	-0.8	12.3	13.4	-1.8	15.1	25.1	-2.3	22.1
Adaptec5	1279.87	155.38	1525.02	156.34	1388.32	158.65	14	-0.8	15.4	9.8	-1.4	10.4	17.2	-2.0	16.6
Newblue1	303.49	46.53	371.98	46.75	332.19	46.88	14.8	-1.1	5.0	7.1	-2.0	2.8	19.5	-3.0	6.8
Newblue2	446.07	75.70	576.44	75.28	441.45	76.36	25.9	-1.8	5.1	10.2	-2.7	3.1	28.5	-3.0	6.5
Average							16.1	-0.9	7.8	9.6	-2.6	10.2	19.9	-2.7	16.67

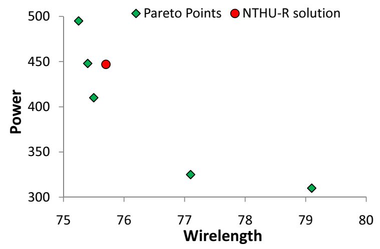


Figure 6: Tradeoff curve for newblue2.

Table 1 shows that by just combining the 3 initial configurations without on-the-fly tree generation (VER1), we can obtain on average 16.1% signal power improvement with only 0.9% wirelength degradation. The average runtime is 7.8 minutes which is extremely fast compared to current routing procedures. If only considering on-the-fly route generation for the last 50K nets (VER2), we also obtain on average 9.6% power improvement with only 2.6% wirelength degradation and on average 10.2 minutes of runtime. Finally, when combining the two cases, we obtain the maximum power saving of 19.9% improvement with average wirelength degradation of 2.7%. Runtime on average increases by 7 minutes.

It is interesting to see if a more relaxed constraint for wirelength degradation and a flexible runtime bound are considered, how much gain our framework is able to provide in terms of signal power saving. We conduct another experiment in which we generate new routes for the last 100K nets and allow 5% wirelength degradation in reference to the wirelength of NTHU-R. Table 3 reports the new results. As can be seen, 2% increase in the tolerable wirelength degradation can provide 6.5% additional improvement in signal power saving when compared to case VER2(50K). This is provided at the expense of on average 10 minutes more run time.

An important strength of our framework is the possibility to provide routing solutions that are optimized both for signal power as well as wirelength (scaled to 10^5). Consider the Pareto curve of Fig. 6. We plot the information for final solutions of benchmark `newblue2`. The solutions are represented as Pareto points in a 2-dimensional space (wirelength vs. power). Our framework was able to generate five different routing solutions for this benchmark, each of which is better than the others in at least one aspect (wirelength or signal power). We also plot the information for the NTHU-R solution of the same benchmark (the red circle in Fig. 6). The solution points are categorized into three different groups. Some Pareto points are the routing solutions that encompass a smaller wirelength, when compared to NTHU-R solution, but a larger signal power. Some others represent routing solutions with less signal power but more wirelength than NTHU-R solution. The points in the third group are corresponding to routing solutions that are optimized for both signal power and wirelength. *So we are able to improve both power and wirelength.* Note that our framework will never generate solutions that are worse than the input solution in the two aspects. This provides a facility to apply any cost function after the computation and select a routing solution that is more appropriate for specific types of applications.

Table 3: Result for 5% tolerable wirelength degradation.

Benchmark	NTHU-R[3]		VER2(100K)		
	P	WL	P(%)	WL(%)	TIME
Adaptec1	475.56	53.44	10.3	-4.0	28.5
Adaptec2	372.13	52.29	18.1	-4.9	17.1
Adaptec3	1085.62	131.01	7.5	-1.9	65.4
Adaptec4	799.37	121.69	13.1	-3.0	47.1
Adaptec5	1279.87	155.38	14.7	-1.8	24.5
Newblue1	303.49	46.53	15.5	-4.8	7.5
Newblue2	446.07	75.70	16.9	-4.6	13.7
Average			13.7	-3.6	29.2

7. CONCLUSIONS

We propose a fast Pareto-algebraic global routing procedure that can significantly improve the signal power. Our model of the signal power includes layer- and size-dependent area and fringe capacitances as well as the coupling capacitance between the wires. Power improvement is achieved by using one or more initially-provided wirelength-optimized routing solutions and by detouring some branches of the provided routes. We also proposed a technique for on-the-fly generation of power-efficient routes within our framework. Our procedure maintains the feasibility of the routing solution in terms of not generating any overflow, and ensures the overall wirelength degradation is not beyond a user-defined parameter, with respect to an initially provided wirelength-optimized solution. In our simulations for the ISPD 2008 benchmarks with machines of only 2GB memory, we show that, with at most 3% tolerable degradation in wirelength, we obtain an average of 19.9% power improvement, when combining the solutions of three state of the art global routing tools with our on-the-fly route generation.

8. ACKNOWLEDGEMENTS

This research is supported by National Science Foundation under award CCF-0914981.

9. REFERENCES

[1] ISPD 2008 [online]http://www.sigda.org/ispd2008/contests/ispd08rc.html.

[2] Nangate 45 nm open cell library, [online] http://www.nangate.com. 2008.

[3] Y.-J. Chang, et al., NTHU - Route 2.0: A fast and stable global router. In *ICCAD*, pages 338–343, 2008.

[4] M. Cho, et al., Boxrouter 2.0: architecture and implementation of a hybrid and robust global router. In *ICCAD*, pages 503–508, 2007.

[5] M. Geilen, et al., An algebra of pareto points. *Fundamenta Informaticae*, 78(1):35–74, 2007.

[6] N. Magen, et al., Interconnect-power dissipation in a microprocessor. In *SLIP*, pages 7–13, 2004.

[7] M. Moser, et al., An algorithm for the multidimensional multiple-choice knapsack problem. *IEICE Trans. on Fundamentals of Electronics*, (E80-A(3)):582–589, 1997.

[8] J. A. Roy and I. L. Markov. High-performance routing at the nanometer scale. In *ICCAD*, pages 496–502, 2007.

[9] H. Shojaei, et al., A parameterized compositional multi-dimensional multiple-choice knapsack heuristic for cmp run-time management. In *DAC*, pages 917–922, 2009.

[10] T.-H. Wu, et al., GRIP: Scalable 3D global routing using integer programming. In *DAC*, pages 320–325, 2009.

[11] Y. Xu, et al., Fastroute 4.0: global router with efficient via minimization. In *ASPDAC*, pages 576–581, 2009.