Escape Routing For Dense Pin Clusters In Integrated Circuits

Muhammet Mustafa Ozdal
Intel Corporation
Hillsboro, OR 97124
mustafa.ozdal@intel.com

ABSTRACT

As the design complexities and circuit densities are increasing, the detailed routing (DR) problem is becoming a more and more challenging problem. Due to the high complexity of DR algorithms, it is very important to start the routing process with clean solutions, rather than starting with suboptimal routes and trying to fix them in iterative process. In this paper, we propose an escape routing algorithm that can optimize routing of a set of nets around their terminals. For this, we first propose a polynomial-time algorithm that guarantees to find the optimal escape routing solution for a set of nets when the track structures are uniform. Then, we use this algorithm as a baseline, and study the general problem with arbitrary track structures. For this, we propose a novel multi-commodity flow (MCF) model that has a one-to-one correspondence with the escape routing problem. This MCF model is novel in the sense that the inter-dependency and contention between different flow commodities is minimal. Using this model, we propose a Lagrangian-relaxation (LR) based algorithm to solve the escape problem. Our experiments demonstrate that this algorithm improves the overall routability significantly by reducing the number of nets that require rip-up and reroute.

Categories and Subject Descriptors

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

General Terms

Algorithms, Design

Keywords

Detailed routing, escape routing, multi-commodity flow

1. INTRODUCTION

As the transistor sizes have been shrinking and the circuit densities have been increasing, the routing problem for integrated circuits (ICs) is becoming a more and more challenging problem. The general problem of routing an arbitrary set of nets using limited routing resources is known to be an NP-complete problem. Furthermore, design for manufacturability (DFM) rules impose complex constraints on the routing problem, making it even harder to devise effective algorithms. Due to the problem complexity, the routing problem is typically solved in two steps: global routing and detailed routing. During global routing, nets are routed on a coarse-grain grid structure with the objective of determining the regions within which each net will eventually be routed. Since a simplified resource model is used during global routing, it is possible to use sophisticated algorithms such as multi-commodity flow [1, 16], or to perform large numbers of rip-up-and-reroute iterations until all capacity constraints in the coarse-grain grid are satisfied [7, 10]. After a rough route is determined for each net, the second step is to perform detailed routing to find the exact routes of all nets. Typically, the grid sizes during detailed routing are much larger than the coarse-grain grids of global routing, and the solution space for individual nets is much larger due to the fine-grain modeling of routing resources. In addition, the complex DFM rules make detailed routing even more difficult, and it becomes prohibitive to use sophisticated routing algorithms with high complexities, which can route all nets simultaneously in a near-optimal way. Typically, detailed routing is performed one net at a time, and the solution space for each net is confined to the regions determined by the global routing.

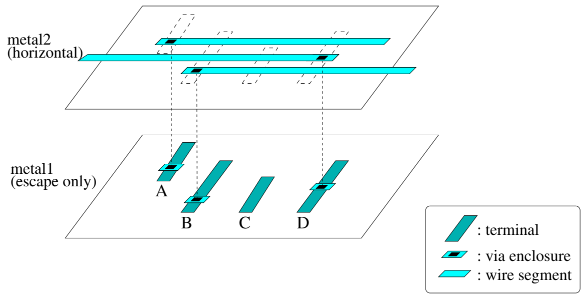


Figure 1: Detailed routing around metal1 terminals of 4 nets is illustrated. Routing is done one net at a time in the order of A-D-B-C, and the terminal of net C is blocked completely.

Due to small flexibility in detailed routing, it is very important to start the process with good routes, since an *undesired route* created in the beginning may have a ripple effect on other nets. Furthermore, the order in which nets are routed is important, and different processing orders may lead to different routing results. As an example, let us consider the detailed routing problem illustrated in Figure 1. Here, adjacent terminals of 4 nets A, B, C, and D are illustrated, together with the detailed routing solution around these terminals. Here, nets have been processed in the order A-D-B-C, and the metal2 segments connected to each terminal are illustrated in the figure. Observe here that, the terminal of net C is blocked by the other routes, and it is not possible to connect it to a metal2 seg-

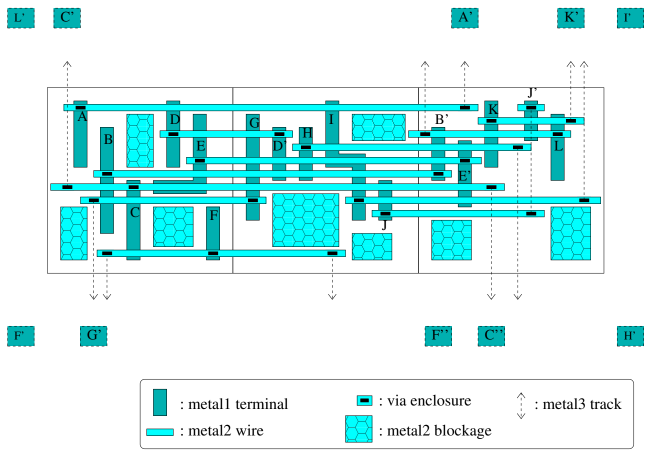


Figure 2: Escape routing solution around a cluster of 16 terminals. The neighboring terminals that are to be routed to these terminals are also shown outside the cluster boundary. Escape routing is performed on metal2 in such a way that routability is maximized and wirelengths are minimized.

ment, which leads to a detailed routing failure. Here, either net B or net D needs to be ripped up for net C to be routed successfully. However, since net B and net D have no other available metal2 resources, their terminals will be blocked unless net A is ripped up, too. As mentioned before, the fine-grain resource modeling and complex design rules make it expensive to perform large number of rip-up-and-reroute iterations in detailed routing. Furthermore, partially ripping up a net is likely to introduce additional bends, and it degrades routing quality. Hence, it is very important to start the routing process with a clean detailed routing solution, rather than starting with suboptimal routes and trying to fix them in an iterative process.

In this paper, we propose an escape routing algorithm to perform detailed routing of a set of nets simultaneously around clusters of terminals. Industrial designs commonly have terminals clustered together in different regions of the circuits due to functional hierarchy and cell placement [14]. Detailed routing around these clusters is more difficult since the routing resources are limited, and multiple nets try to compete for these limited resources. Making a suboptimal decision here is likely to lead to more routing failures, as shown in the example of Figure 1. On the other hand, starting with a close-to-optimal routing solution within these clusters can greatly simplify the rest of the routing. As an example, let us consider the escape routing solution in Figure 2. Here, 12 nets (A-L) are illustrated with 16 terminals inside a cluster. The relative locations of the neighbors of these terminals are also illustrated in this figure. For example, terminal A has a neighbor A' outside the cluster, which aligns with the cluster boundary in the vertical orientation. Conversely, the terminals B and B' are both within the cluster, and they need to be connected to each other with a metal2 segment. The nets with more than 2 terminals (e.g. terminals F, F', and F'') can also be handled within this framework. The objective here is to escape route as many nets as possible, while maximizing routability and minimizing the total wire lengths. In particular, we determine the metal2 segments of a set of nets around metal1 terminal clusters in such a way that the overall detailed routing problem is simplified. For example, in Figure 2, a metal2 segment is connected to terminal A such that it aligns with the other terminal A'. Similarly, since both terminals B and B' are in the cluster, a metal2 segment is created connecting these two terminals. On the other hand, the metal2 segment of terminal F could not be extended to align with terminals F' and F'' due to lack of routing resources; however, the connection is made so as to minimize the misalignment. Conversely, if we perform detailed routing one net at a time around this cluster (as opposed to routing them simultaneously), the metal2 segments of the nets routed earlier will be likely to block the terminals of the nets routed later.

Escape routing is a well-studied problem especially in the package routing domain [13, 14, 15]. Typically, network flow based algorithms are used to route multiple nets simultaneously from their pins to the cluster boundaries. However, these algorithms have the shortcomings that it is hard to impose constraints on individual routes such as alignment and length constraints (e.g. the escape route of terminal A aligns with the neighboring terminal A' in Figure 2); it is hard to handle intra-cluster connections (e.g. terminals B and B' are connected to each other within the cluster); and it is hard to optimize non-linear cost functions (see Section 2). Another related problem is the track assignment problem [2], which tries to assign a set of routing segments on a set of available routing tracks. However, the problem in this paper is different in the sense that the lengths and spans of metal2 segments are not fixed, and they need to be optimized while determining the track assignment. For example, the metal2 segment of terminal L could not be aligned with terminal L' in Figure 2. Assigning a longer metal2 segment to terminal L would block one of the terminals: B, E, or H. Hence, it is important to determine the lengths and spans of metal2 segments based on the optimal solution for a set of nets, rather than for a single net.

The rest of the paper is organized as follows. In Section 2, we provide a formal description of the escape problem. In Section 3, we propose a polynomial-time optimal algorithm to solve the escape problem for terminal clusters with uniform track structures. Then, we study the general problem with non-uniform tracks in Section 4. We propose a fast Lagrangian relaxation (LR)-based multi-commodity flow algorithm to solve this problem effectively. Our formulation is novel in the sense that each flow represents one routing track instead of one net. This formulation has the advantage that resource contention between different flows is small compared to a general multi-commodity flow formulation. This enables us to propose a fast and efficient LR-based algorithm to solve the problem. Finally, we demonstrate the effectiveness of our models and algorithms through experiments in Section 5.

2. PROBLEM FORMULATION

Assume that a cluster of terminals $\mathcal{T}$ is identified on one layer of a circuit, where each layer has either horizontal or vertical preferred routing orientation. Without loss of generality, assume that the terminals in the cluster are on the bottom layer (metal1), which has a vertical orientation. Our objective here is to find an escape routing solution for all terminals in $\mathcal{T}$ on the layer above them (metal2) such that routability is maximized and wire lengths are minimized.

To realize these objectives, escape routing needs to be done in accordance with the topology of the nets corresponding to these terminals. Figure 3 illustrates the shortest and the longest metal2 escape segments for terminals A, B, and C, which belong to different nets with different topologies[1]. For example, in part (a), terminal A has two neighbors to its right. Hence, it can have a horizontal metal2 segment that can extend until the rightmost neighbor. On the other hand, in the topology shown in part (b), terminal B has

[1]These topologies can be determined based on the minimum Steiner trees or global routing solutions.

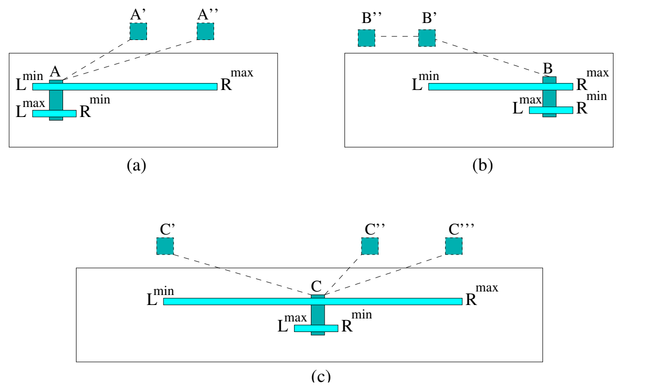


Figure 3: The feasible ranges for the escape segments of terminals A, B, and C are illustrated based on the net topologies.

a single neighbor, and it is not necessary to extend the metal2 segment to the left beyond this neighbor. Similarly, terminal C in part (c) has two neighbors on both sides, and the metal2 segment can be extended to the left and right. While the maximum spans of the escape segments are determined based on net topologies, the minimum spans can be determined based on electrical requirements and design rules such that connection to the metal2 segment is possible from other layers.

Based on this, it is possible to define a feasible range for the left and right metal2 segments of each terminal as follows. Let L_T (R_T) denote the leftmost (rightmost) coordinate of the metal2 segment above terminal T. Let L_T^{min} (R_T^{min}) and L_T^{max} (R_T^{max}) denote the minimum and maximum feasible coordinates for L_T (R_T), respectively. Figure 3 illustrates how to determine these values based on net topologies. Furthermore, it is possible to assign different preference values for different metal2 segment configurations of a terminal. Let $g_T(L_T)$ ($g_T(R_T)$) be a preference function that indicates how desirable a particular L_T (R_T) value is for a terminal.

We can formulate the escape routing problem as follows: Given a set of available metal2 tracks, and a set of terminals $\mathcal{T}$ on metal1, find L_T ($L_T^{min} \leq L_T \leq L_T^{max}$) and R_T ($R_T^{min} \leq R_T \leq R_T^{max}$) values for each $T \in \mathcal{T}$ such that the expression $\sum_{T\in\mathcal{T}} g_T(L_T) + \sum_{T\in\mathcal{T}} g_T(R_T)$ is maximized, and such that each metal2 segment $[L_T, R_T]$ for $T \in \mathcal{T}$ can be assigned to an available metal2 track on top of T. In practice, it is desirable for the preference function to reflect the fact that we want to obtain escape segments that have lengths as close as possible to L_T^{max} and R_T^{max} values so that alignment with neighbors is maximized. However, it is also desirable to obtain a balanced solution with more nets successfully escaping, instead of few long escape segments occupying the whole tracks. In our experiments we set the preference function to be $g(L) = \sqrt{L}$, where L is the length of the escape segment[2]. However, our proposed algorithms can optimize any given objective function.

3. OPTIMAL ALGORITHM FOR UNIFORM TRACKS

In this section, we propose an optimal algorithm for the cases where the track structures are uniform, and all terminals are aligned with each other. This restriction in the input problem lets us represent all available tracks as a single lumped routing resource, and it enables us to propose an optimal algorithm. We will use this algorithm as a baseline for the general algorithm we propose in Section 4, where there is no restriction for the uniformity of the tracks. Figure 4(a) illustrates an example where all available tracks (shown as dashed lines) are identical, and all terminals align with each other.

[2]For internal connections (e.g. terminals B and B' in Figure 2, the preference function is set to be a step function that has nonzero value only for the length that realizes the internal connection.

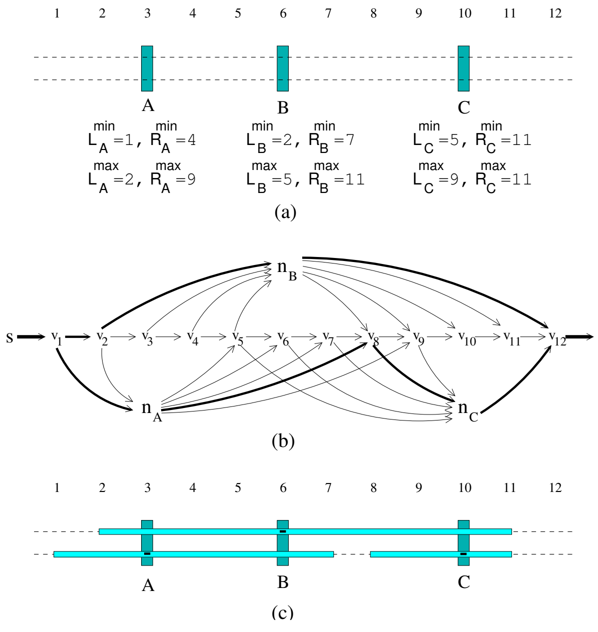

1 2 3 4 5 6 7 8 9 10 11 12
A
B
C
L_A^{min}=1, R_A^{min}=4
L_A^{max}=2, R_A^{max}=9
L_B^{min}=2, R_B^{min}=7
L_B^{max}=5, R_B^{max}=11
L_C^{min}=5, R_C^{min}=11
L_C^{max}=9, R_C^{max}=11
(a)
n_B
s
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11} v_{12}
d
n_A
n_C
(b)
1 2 3 4 5 6 7 8 9 10 11 12
A
B
C
(c)

Figure 4: (a) A sample problem with track bounds specified for each terminal based on net topologies (not shown in the figure). (b) The corresponding min-cost flow solution highlighted. (c) The corresponding escape routing solution on 2 metal2 tracks.

Assume that the minimum and maximum horizontal coordinates of the given terminal cluster are x_{min} and x_{max}, respectively. The algorithm we propose is based on the flow network given in the following definition.

Definition 3.1. *The flow network $\mathcal{N}$ corresponding to a given cluster of terminals $\mathcal{T}$ is created as follows:*

- *A vertex v_x exists in $\mathcal{N}$ for each x in the interval $[x_{min}, x_{max}]$ with infinite node capacity. Similarly, a vertex n_T exists for each terminal T with capacity 1.*
- *A source vertex s and a drain vertex d exist in $\mathcal{N}$ with node capacities equal to the number of available tracks.*
- *An edge exists in $\mathcal{N}$ from v_x to v_{x+1}, for each $x_{min} \leq x < x_{max}$. This edge is defined to have zero cost.*
- *Let T be a terminal with track bounds L_T^{min}, L_T^{max}, R_T^{min}, R_T^{max}, and a preference function $g(x)$, as defined in Section 2. An edge exists in $\mathcal{N}$ from each v_x ($L_T^{min} \leq x \leq L_T^{max}$) to n_T, if there is no blockage between track coordinate x and the terminal T. The cost of this edge is defined to be $-g(x)$. Similarly, an edge exists from n_T to each v_{x+1} ($R_T^{min} \leq x \leq R_T^{max}$), if there is no blockage between the terminal and track coordinate x. The cost of this is edge is defined to be $-g(x)$.*
- *An edge exists in $\mathcal{N}$ from source vertex s to $v_{x_{min}}$ with zero cost. Similarly, an edge exists from $v_{x_{max}}$ to drain vertex d with zero cost.*

Figure 4(b) illustrates the flow network corresponding to the problem in part (a). Based on this flow network definition, we propose the algorithm given in Figure 5 to solve the escape routing

```
ESCAPE-ROUTE-UNIFORM-TRACKS
  Create a flow network $\mathcal{N}$ as defined in Definition 3.1.
  Solve the min-cost flow problem on $\mathcal{N}$
  Consider each vertex $n_T$ that has nonzero flow passing through
    it. Due to flow conservation constraint, there must be exactly
    one vertex $v_{x_L}$ and one vertex $v_{x_R}$ such that there is a
    unit flow through edge $v_{x_L} \to n_T$ and a unit flow
    through edge $n_T \to v_{x_R}$. Create a metal2 segment
    between $x_L$ and $x_R - 1$, and add it into segment set $\mathcal{S}$.
  Use left-edge algorithm [8] to assign the metal2 segments in $\mathcal{S}$
    to the available tracks in the cluster.
```

Figure 5: Algorithm to optimally solve the escape routing problem for uniform track structures.

problem. Here, we first solve the min-cost flow problem optimally using an existing polynomial-time algorithm [11]. In Figure 4(b), the total flow from s to t is equal to 2, which is the number of available tracks in the problem. The min-cost flow in this network corresponds to the escape routing solution that maximizes the total preference value. Figure 4(c) illustrates the final escape routing solution for the problem given in part (a). In this figure, it is possible to observe the one-to-one correspondence between the network flow and the escape routing solution.

Theorem 3.1. *The algorithm given in Figure 5 finds the optimal escape routing solution for a cluster of terminals with uniform track structures.*

PROOF. There is a one-to-one correspondence between the feasible metal2 segments and the network flow model defined in Definition 3.1. Furthermore, the maximum flow from source node s to drain node d is equal to the number of available tracks due to the capacity constraints of nodes s and t. Hence, the number of metal2 segments selected that overlap with any x coordinate (i.e. channel density) must be less than or equal to the number of tracks. It is well known that track assignment of a set of fixed segments can be solved optimally in linear time using left-edge algorithm [8] iff the channel density is less than or equal to the number of available tracks. Further details of the proof are omitted due to page limitations. □

4. GENERALIZED ALGORITHM FOR NONUNIFORM TRACKS

In this section, we generalize our models to be able to solve the escape routing problem for arbitrary track structures. For this purpose, we first propose an exact model that is based on multi-commodity flow (MCF). Although MCF is an NP-complete problem, a special property in our model enables us to propose an efficient Lagrangian-relaxation based algorithm.

4.1 Exact Model Based on MCF

Let x_{min} and x_{max} denote the minimum and maximum horizontal coordinates of the given terminal cluster, and let t_{min} and t_{max} denote the minimum and maximum available tracks.

Definition 4.1. *The multicommodity flow network $\mathcal{N}$ corresponding to a given cluster of terminals $\mathcal{T}$ is created as follows:*

- *A distinct flow type f_t is defined corresponding to each track t, $t_{min} \leq t \leq t_{max}$.*
- *A vertex $v_{x,t}$ exists in $\mathcal{N}$ ($\forall x, x_{min} \leq x \leq x_{max}$ and $\forall t, t_{min} \leq t \leq t_{max}$) with capacity 1 for flow type f_t, and capacity 0 for all other flow types. Similarly, a vertex n_T exists for each terminal T, with common capacity 1 for all flow types.*
- *A source vertex s and a drain vertex d exist in $\mathcal{N}$ with node capacities equal to the number of available tracks.*
- *An edge exists in $\mathcal{N}$ from $v_{x,t}$ to $v_{x+1,t}$ ($\forall x, x_{min} \leq x < x_{max}$ and $\forall t, t_{min} \leq t \leq t_{max}$) with zero cost.*
- *Let T be a terminal with track bounds L_T^{min}, L_T^{max}, R_T^{min}, R_T^{max}, and a preference function $g(x)$, as defined in Section 2. An edge exists in $\mathcal{N}$ from each $v_{x,t}$ ($L_T^{min} \leq x \leq L_T^{max}$, and $t_{min} \leq t \leq t_{max}$) to n_T iff a metal2 segment on track t can be created between coordinate x and terminal T. The cost of this edge is defined to be $-g(x)$. Similarly, an edge exists from n_T to each $v_{x+1,t}$ ($R_T^{min} \leq x \leq R_T^{max}$ and $t_{min} \leq t \leq t_{max}$) iff a metal2 segment on track t can be created between terminal T and coordinate x. The cost of this is edge is defined to be $-g(x)$.*
- *An edge exists in $\mathcal{N}$ from source vertex s to each $v_{x_{min},t}$ ($t_{min} \leq t \leq t_{max}$) with zero cost. Similarly, an edge exists from each $v_{x_{max},t}$ ($t_{min} \leq t \leq t_{max}$) to drain vertex d with zero cost.*

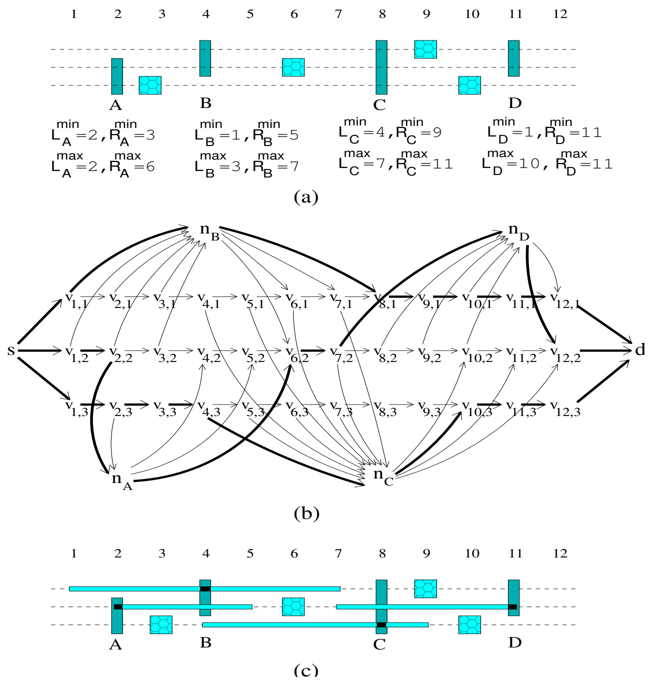
A sample escape problem is illustrated in Figure 6(a) for a cluster of 4 terminals, where the min/max track bounds are specified as input for each terminal. Here, some tracks are not available for some terminals due to the existence of blockages, or misalignment of terminals with some tracks. The corresponding flow network is illustrated in part (b). Observe that the existence of blockages limits the edges entering to and exiting from the nodes corresponding to the terminals. For example, due to the blockage in the second track, the edges from $v_{4,2}$, $v_{5,2}$, $v_{6,2}$ to n_C do not exist, while the corresponding edges exist in the first and third tracks. Although not explicitly marked in the figure, there are 3 flow types f_1, f_2, and f_3 corresponding to each track; only flow type f_t can pass through a vertex $v_{x,t}$ due to capacity constraints given in Definition 4.1. The following analysis shows that there is a one-to-one correspondence between the min-cost flow problem on this network and the escape routing problem.

Lemma 4.1. *If there is a non-zero flow passing through node n_T (corresponding to terminal T), then there must be two nodes $v_{x_L,t}$ and $v_{x_R,t}$ (corresponding to different coordinates of the same track t), such that there is unit flow passing through $v_{x_L,t} \to n_T \to v_{x_R,t}$.*

PROOF. According to Definition 4.1, only one type of flow (f_t) can pass through any node $v_{x,t}$. So, if flow f_t enters n_T from $v_{x_L,t}$, then it has to exit to another node $v_{x_R,t}$ on the same track due to flow conservation constraint. □

Theorem 4.2. *There is a one-to-one correspondence between the min-cost flow problem for the network defined in Definition 4.1 and the escape routing problem as defined in Section 2.*

PROOF. A similar mapping can be used as in Figure 5 to create the metal2 segments from the min-cost flow solution. Figure 6(c) illustrates the escape routing solution corresponding to the flow in part (b). For example, there is a flow $v_{1,1} \to n_B \to v_{8,1}$ in part (b), and a metal2 segment exists in part (c) between coordinates 1 and 7 of the first track. From Lemma 4.1, we know that only one type of flow (corresponding to one track) can pass through a terminal node in the flow solution. Furthermore, each $v_{x,t}$ has a capacity of 1 for only commodity f_t; hence there cannot be a conflict between metal2 segments selected on the same track. Since the edge costs in the network are set as the negatives of the preference values of the corresponding metal2 segments, the min-cost flow solution corresponds to the escape solution with maximal total preference. Further details of the proof are omitted due to page limitations. □



1 2 3 4 5 6 7 8 9 10 11 12
A
B
C
D
$L_A^{min}=2, R_A^{min}=3$
$L_A^{max}=2, R_A^{max}=6$
$L_B^{min}=1, R_B^{min}=5$
$L_B^{max}=3, R_B^{max}=7$
$L_C^{min}=4, R_C^{min}=9$
$L_C^{max}=7, R_C^{max}=11$
$L_D^{min}=1, R_D^{min}=11$
$L_D^{max}=10, R_D^{max}=11$
(a)
n_B
n_D
s
d
n_A
n_C
(b)
(c)

Figure 6: (a) An escape routing problem, where patterned rectangles represent blockages on metal2. (b) The corresponding min-cost flow highlighted. (c) The corresponding escape routing solution.

It is known that multi-commodity network flow is an NP-complete problem, and even the approximation algorithms are typically too expensive to be applied on detailed routing problems. However, due to a special property in our network flow model, we propose an effective Lagrangian-relaxation based algorithm to solve this problem.

4.2 Lagrangian-Relaxation Based Algorithm

Compared to the traditional multi-commodity flow (MCF) model for general routing problems [1, 16], the model we propose has two important advantages. First, the number of commodities (distinct flow types) is equal to the number of tracks, as opposed to the number of nets. Secondly, the flow contention is possible only at the nodes corresponding to the terminals, as opposed to every individual routing resource. In a typical MCF formulation of routing, multiple commodities (corresponding to nets) compete for capacity constrained nodes/edges (corresponding to shared routing resources). However, in our model, the flow contention is possible only between the nodes corresponding to terminals (n_T), instead of the nodes corresponding to individual routing resources ($v_{x,t}$). Hence, the resource contention is significanly lower in our model, making the flow computations easier. Furthermore, we can state the following theorem.

Theorem 4.3. *The multi-commodity min-cost flow problem for network $\mathcal{N}$ defined in Definition 4.1 is equivalent to the following problem: Find a set of min-cost paths from $v_{x_{min},t}$ to $v_{x_{max},t}$ for each track t ($t_{min} \leq t \leq t_{max}$) under the following constraint: The number of paths passing through each vertex n_T corresponding to each terminal T ($T \in \mathcal{T}$) is at most one.*

PROOF. From Definition 4.1, a unit flow of type f_t can only pass through nodes $v_{x,t}$ corresponding to track t, and nodes n_T corresponding to terminals. Hence, the only possible conflicts between different commodities are possible at the nodes corresponding to terminals. □

```
ESCAPE-ROUTE-NONUNIFORM-TRACKS
  Initialize all LM values to 0
  Create a network $\mathcal{N}$ as defined in Definition 4.1
  while convergence condition not occured do
    find the min-cost path for each track $t$
    udpate the LM values (Eq 4) and the edge costs (Eq 3)
  Map the edges in min-cost paths to metal2 segments
```

Figure 7: LR-based algorithm for generalized escape routing

Lagrangian relaxation (LR) is a general technique for solving optimization problems with difficult constraints. The main idea is to replace each complicating constraint with a penalty term in the objective function. Specifically, each penalty term is multiplied by a constant called Lagrangian multiplier (LM), and added to the objective function. The Lagrangian problem is the optimization of the new objective function, where difficult constraints have been relaxed and incorporated into the new objective function. If the optimization is a minimization problem, then the solution of the Lagrangian problem is guaranteed to be a lower bound for the original optimization. So, the objective is to find the best LM values such that the optimal value obtained for the Lagrangian problem is as close to the real optimal value as possible. For this purpose, the LM values are updated iteratively (typically using subgradient method) in the high level, while the relaxed Lagrangian problem is solved for the fixed LM values in low level iterations. Further details about LR can be found in [5, 6].

The escape routing problem as defined in Theorem 4.3 can be formulated as the following constrained optimization problem. Find the set of min-cost paths $\mathcal{P}$ with the following objective:

$$\begin{aligned} &minimize \qquad \sum_{e \in \mathcal{P}} cost(e) \\ &subject\ \ to: \\ &\qquad\qquad \forall T, \quad f(n_T) \leq 1 \end{aligned} \tag{1}$$

Here, $f(n_T)$ denotes the total flow passing through node n_T corresponding to terminal T. Intuitively, it corresponds to the number of tracks on which terminal T has an escape routing solution. If we apply LR on this formulation, our objective becomes the minimization of:

$$\sum_{e \in \mathcal{P}} cost(e) + \sum_{T \in \mathcal{T}} \lambda_T (f(n_T) - 1) \tag{2}$$

Here, λ_T is the LM value corresponding to terminal T. The optimization of this objective function is equivalent to solving a min-cost path problem from each $v_{x_{min},t}$ to $v_{x_{max},t}$ when the edge costs in $\mathcal{N}$ are set as follows:

$$LRcost(e) = cost(e) + \begin{cases} \lambda_T (f(n_T) - 1) & \text{if } n_T \in e \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The LR-based algorithm we propose for the escape routing problem is given in Figure 7. Here, we first solve the min-cost path problems without considering the capacity constraints given in (1). It is known that min-cost paths can be found in linear time for acyclic graphs [3]. After the first iteration, it is possible that some nodes corresponding to terminals have flow values greater than 1. Intuitively, if a node n_T has more than 1 flow, it means that terminal T will have more than 1 metal2 segments (on different tracks) assigned to it. To prevent this, we update the edge costs as in (3), while gradually increasing the λ_T values corresponding to terminals that violate the constraints. As the costs of the violating edges are increased gradually, min-cost paths will pass through other terminal nodes (possibly with smaller preference values), instead of the terminals with multiple flows. Given a flow solution in iteration k, and the current multiplier values λ_T^k, the multipliers for iteration $k+1$ are calculated as follows:

$$\lambda_T^{k+1} = max(0, \lambda_T^k + t_k(f(n_T) - 1)) \tag{4}$$

Here, we use an update schedule similar to subgradient method. Note that t_k is the step size used in subgradient method, and it is updated in each iteration such that it slowly converges to zero. Specifically, we use the convergence condition given by Held, et al [9], which states that as $k \to \infty$, it should be the case that $t_k \to 0$ and $\sum_{i=1}^{k} t_i \to \infty$. In particular, we have used $t_k = 1/k^\alpha$ in our experiments, where $\alpha < 1$ is a constant to provide a trade-off between solution quality and convergence speed. In our experiments, we have set α to 0.1.

5. EXPERIMENTAL RESULTS

We have implemented our algorithms in C++, and performed experiments on a computer with Intel Xeon 3.6Ghz processor and Linux operating system. Since the objective of escape routing is to make the detailed routing (DR) stage easier, we have tested our algorithms within a DR framework, which routes nets one by one, and uses history-based rip-up and reroute [4, 12] to resolve resource contention between different nets.

Table 1 illustrates our results on a set of test circuits, all of which have arbitrary track structures. Here, two sets of experiments have been performed. In the first set, DR is applied directly to route the nets. In the second set, escape routing is performed on each cluster of terminals before the actual DR algorithm. In this table, "pre-rnr opens" denotes the number of nets that could not be routed (due to conflicts with other nets) in the first iteration, before the rip-up and reroute (RNR) stage. We also report the average number of vias per net ("via avg"), the number of nets that could not be routed after 10 iterations of RNR ("final opens"), and the total execution times. The wirelength results obtained in the experiments were within 1% difference; hence they are not reported.

As discussed earlier, nets are routed one by one in DR, and the nets routed earlier possibly block the terminals of the nets routed later. On the other hand, our escape routing algorithm simultaneously allocates routing resources for all terminals in a cluster, preventing other nets to block these terminals during DR. The results in Table 1 illustrate that the number of nets that need RNR is reduced by 64% when escape routing is performed before DR. Note that performing RNR on a large set of nets during DR is usually expensive, and it may lead to suboptimal results. Reducing the number of open nets before RNR reduces the number of final opens by 78%, improves the average via counts per net by 4%, and decreases the total execution times by 34%, as shown in the table.

6. CONCLUSIONS

In this paper, we propose escape routing algorithms for dense pin clusters in integrated circuits. We first propose an optimal algorithm based on network flow for uniform track structures. Then, we propose a Lagrangian relaxation based algorithm to generalize the problem domain to arbitrary track structures. Our experiments demonstrate that the proposed escape routing methodology can reduce the number of nets that require rip-up and reroute significantly, through simultaneous routing resource allocation for all terminals within a dense cluster.

Table 1: Experimental results on test circuits

			TRADITIONAL DR				ESCAPE + DR			
Ckt	Grid size	Net cnt	pre-rnr opens	via avg	final opens	time (m:s)	pre-rnr opens	via avg	final opens	time (m:s)
D1	1.5M	12K	3382	5.52	319	18:28	1339	5.16	30	9:58
D2	3M	18K	4322	4.82	55	25:55	1474	4.60	6	16:31
D3	4M	22K	5747	4.80	168	20:38	2186	4.56	46	13:34
D4	6M	29K	5781	4.42	67	25:22	1904	4.27	26	18:41
D5	7.5M	36K	8179	4.71	151	32:18	2760	4.53	48	21:40
D6	9M	48K	11938	4.87	324	44:16	4124	4.67	73	28:30
D7	12M	56K	10493	4.69	203	51:27	4196	4.52	53	35:01
Avg			1.00	1.00	1.00	1.00	0.36	0.96	0.22	0.66

7. REFERENCES

[1] C. Albrecht. Provably good global routing by a new approximation algorithm for multicommodity flow. In *Int'l Symposium on Physical Design*, pages 19–25, 2000.

[2] S. Batterywala, N. Shenoy, W. Nicholls, and H. Zhou. Track assignment: a desirable intermediate step between global routing and detailed routing. In *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design (ICCAD)*, Nov. 2002.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1992.

[4] C. Ebeling, L. McMurchie, S. A. Hauck, and S. Burns. Placement and routing tools for the triptych FPGA. *IEEE Trans. on VLSI*, pages 473–482, 1995.

[5] M. L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

[6] M. L. Fisher. An applications oriented guide to lagrangian relaxation. *Interfaces*, 15(2):10–21, 1985.

[7] R. T. Hadsell and P. H. Madden. Improved global routing through congestion estimation. In *Proc. ACM/IEEE Design Automation Conf.*, 2003.

[8] A. Hashimoto and J. Stevens. Wire routing by optimizing channel assignment within large apertures. In *Proc. of Design Automation Workshop*, pages 155–169, 1971.

[9] M. H. Held, P. Wolfe, and H. D. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6(1):62–88, 1974.

[10] R. Kastner, E. Bozogzadeh, and M. Sarrafzadeh. Predictable routing. In *Proc. IEEE/ACM Intl. Conf. Computer-Aided Design*, 2000.

[11] J. Orlin. A faster strongly polynomial minimum cost flow algorithm. In *Proc. of ACM Symp. on Theory of Computing*, 1988.

[12] M. M. Ozdal and M. D. F. Wong. A length-matching routing algorithm for high-performance printed circuit boards. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 25:2784–2794, 2006.

[13] M. M. Ozdal and M. D. F. Wong. Algorithms for simultaneous escape routing and layer assignment of dense PCBs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, Aug. 2006.

[14] M. M. Ozdal and M. D. F. Wong. Optimal routing algorithms for pin clusters in high-density multichip modules. In *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design (ICCAD)*, Nov. 2005.

[15] R. Shi and C. C.-K. Efficient escape routing for hexagonal array of high density I/Os. In *Proc. of Design Automation Conf.*, 2006.

[16] E. Shragowitz and S. Keel. A global router based on a multicommodity flow model. *Integration: the VLSI*, 5(1):3–16, 1987.