Track Assignment: A Desirable Intermediate Step Between Global Routing and Detailed Routing

Shabbir Batterywala, Narendra Shenoy, William Nicholls and Hai Zhou[1]
{battery,nshenoy, nicholls}@synopsys.com, haizhou@ece.northwestern.edu
Synopsys Inc.
700, East Middlefield Road
Mountain View, CA 94043.

Abstract— **Routing is one of the most complex stages in the back-end design process. Simple routing algorithms based on two stages of global routing and detailed routing do not offer appropriate opportunities to address problems arising from signal delay, cross-talk and process constraints. An intermediate stage of track assignment between global and detailed routing proves to be an ideal place to address these problems. With this stage it is possible to use global routing information to efficiently address these problems and to aid the detailed router in achieving the wiring completions. In this paper we formulate routing as a three stage process; global routing, track assignment and detailed routing. We describe the intermediate track assignment problem and suggest an efficient heuristic for its solution. We introduce cost metrics to model basic effects arising from connectivity. We discuss extensions to include signal integrity and process constraints. We propose a heuristic based on weighted bipartite matching as a core routine. To improve its performance additional heuristics based on lookahead and segment splitting are also suggested. Experimental results are given to highlight the efficacy of track assignment stage in routing process.**

I. Introduction

As VLSI technology reaches deep submicron (DSM) dimensions and gigahertz clock frequencies, interconnect has become the dominant factor in determining performance, power, and reliability of a system [1]. Even though some optimizations such as buffering, wire sizing, and wire spacing can be done after wire routing to improve system performance, routing is still the most important step to do the optimizations or make them feasible during later stages. Besides wiring completion, a DSM router should also consider wire delays, signal integrity, and reliability issues. Designing a router with many conflicting requirements is a big challenge.

Traditionally, a router usually consists of two stages: *global routing* and *detailed routing* [2]. In the global routing stage, the whole routing region is divided into an array of rectangular subregions, each of which may accommodate tens of routing tracks in each dimension. These subregions are usually called *global cells*. Pins of nets are then mapped to subregions, and routing resources and obstructions are modeled as capacities on boundaries of subregions. For each net, a route is constructed as a set of rectangular subregions within which a physical connection can be embedded. Usually Steiner tree and maze search techniques are used here to generate a global routing. In the detailed routing stage, a physical connection is constructed for each net within the global routing cells determined in the previous step. Since detailed routing must deal with geometries of design objects (pins, pre-routes and obstructions) and different design rules (metal spacing, metal width etc.) confining the connection within given global routing regions greatly prunes the search space and speeds up the search time.

There are generally two approaches to do detailed routing. One is to use a sequential area router to connect each net at a time (see [3]). The global route for the concerned net is used as a guide to constrain the detailed route to be within the given region. The other approach is to partition the problem into a sequence of switch-box routings on each global cell. For each net crossing a global cell, a set of cross-points is determined dictated by the global routing. This will fix the position through which a net crosses a given global cell boundary. Tseng et al [4] and Chang et al [5] study the cross point assignment problem to optimize crosstalk noise. Tseng et al [4] address interconnect delay in their optimization problem. Even though they both try to straighten a route by aligning the cross points of the same net, that goal may not be fully achieved since crosstalk or interconnect delay is given the highest priority. Also, since layers are already fixed during global routing, the option to assign nets to different layers to avoid crosstalk is not available.

Detailed routing stage usually takes a long time searching entire subregions given by the global router. Also observe that the maze search in the detailed routing is done in a sequential net-by-net fashion. Consequently there exist paths with many detours even though the global route is straight. Paths with detours consume more routing resources and in many cases reduce routability of other nets, specially in congested regions. Routability problems can be addressed with a smart rip-up and re-route stage. However it is a time consuming solution. Based on these observations, we strongly believe that it is important to integrate a track (and layer) assignment between global and detailed routing. We propose to assign long global routes, *i.e.,* routes running over at least one whole global cell, to underlying routing resources. We call these long paths as *iroutes* (as a short name for *interval of global route*). The routing resources are present in the form of gridlines and vias. We call these gridlines as *tracks*. They represent candidate locations where iroutes can be assigned.

The first benefit of such a track assignment is that it fully utilizes the information generated by the global router. With a negligible fraction of the original detailed router's running time, it embeds a large part of the routes. The second benefit is the efficient usage of routing resources since most of the routes are laid in straight lines. The third benefit is that track assignment introduces a stage where nets are routed in parallel. This becomes especially important when both global and detailed routing route nets in a sequential manner. Lastly, since a significant portion of all the global routes are realized as iroutes to be assigned on the tracks, the neighborhood information for large nets can be easily calculated. Thus track assignment provides a suitable stage to optimize DSM issues that require neighborhood information.

Track assignment has been a well studied problem from vari-

[1]This work was done when Hai Zhou was employed with Synopsys Inc. He is currently with ECE Dept. Northwestern University, Evanston, IL 60208

0-7803-7607-2/02/$17.00 ©2002 IEEE

ous aspects. The first application of track assignment can be found in the context of constrained via minimization. Kuo et al [6] focus on the two layer problem. Extensions to multi-layer are provide by Shi [7] and Chang and Cong [8]. These approaches improve an existing routing of a design, while we are concerned about constructing a routing. Track assignment has found use in Multi-Chip Module (MCM) routing although the constraints are somewhat different. Sriram and Kang [9] and Ho et al [10] discuss the track assignment problem in this context. The freedom of track assignment has also been utilized to improve the delay of global interconnects by Ciesielski [11] and Saxena and Liu [12]. Cho et al [13] assign cross-talk sensitive nets to different layers, while simultaneously minimizing the number of vias and layers. Cong et al [14] have proposed a three stage routing system which performs global routing, wire planning and detailed routing for variable width and variable spacing designs. The wire planning stage is very similar to the track assignment which we do. Though, the focus during wire planning is only congestion minimization and is done on a net by bet basis. Obviously the quality of wire planning is dependent on net ordering. We however believe that in order to best address the DSM problems all nets should be planned simultaneously. An independent work similar to our proposed track assignment has been described by Kay et al [15]. However, their work is motivated only by crosstalk optimization. It does not consider routability issues such as pin positions and obstructions. Formulating crosstalk constraints as forbidding pairs of wires to be assigned to adjacent tracks, they use an integer linear programming (ILP) to solve the assignment problem. We also draw the attention of the readers to U.S. patent by Groeneveld and van Ginneken [16] which describes a system similar to the one we propose. However none of the references in literature or the patent describe the issues and impact of incorporating track assignment in a routing flow. Our approach verifies that the incorporation such a step in the routing flow is possible without any undue burden on the quality of results.

The rest of the paper is organized as follows. In Section II we briefly describe our routing system. In Section III, we will discuss different possible requirements on track assignment. Its subsections contain, problem formulation (Section III-A), cost metrics to model the desirability of assigning iroutes to tracks (Section III-B), underlying graph model for the track assignment problem (Section III-C), an algorithm based on weighted bipartite matching as a sub-optimal solution for the problem (Section III-D), a look-ahead heuristic to improve the performance of the algorithm (Section III-E) and and iroute splitting scheme to reduce the detailed routing times (Section III-F). In Section IV, we present some experimental results. We conclude the paper in Section V.

II. Brief Overview of Our Routing System

We now give a very brief outline of the routing system which we have developed. We number the routing layers (also called *planes*) from zero upwards, layer zero being the first layer of metal. On a plane we model routing resources as a set of parallel gridlines. Gridlines on a plane can be either horizontal or vertical, depending on the preferred routing direction of that plane. They can be non-uniformly spaced on a plane and can be spaced differently on different layers. Even though the gridlines are either vertical or horizontal on a plane, connections perpendicular to them are allowed at a slightly higher cost. These connections are called *jogs*. The routing process starts with a netlist which needs to be routed. Each net in the netlist is a bag of components, such as pins and pre-routes (called *netcs*) that are to be connected together. The geometry of these netcs, and routing obstructions, etc. is first read and stored in the routing database. These geometries are mapped onto intervals on gridlines. The database facilitates all the necessary queries and provides for efficient manipulation of these intervals. The routing process begins with global routing wherein the routing region is partitioned into global cells (GCs). A global route is built over the grid graph of GCs for each net. It consists of a set of paths comprising of GCs. It defines a region in which a metal path has to be constructed to complete routing of the net. Post global routing, we do track assignment wherein contiguous rows (columns) on the global route paths are identified and space is reserved for them on the underlying gridlines in these rows (columns). We will discuss more about the track assignment in subsequent sections. Since we use a two-dimensional model for global routing, we use the track assignment step to also obtain a layer assignment. Post track assignment, we make the final connections to connect all the netcs and iroutes of a net in the detailed routing stage. After one pass of detail routing with no rip-up, we perform a few iterations of rip-up and re-route. Iroutes are ripped up with a higher cost than the metal segments created by detail routing. If an iroute is ripped up, then the GCs that led to the creation of the iroute are used by the detailed router to find the connections.

III. Track Assignment

An objective for track assignment is that it should be fast and yet provide a good starting point for the detail router to complete with relative ease. To do so we free the track assigner from tedious details of local connections. We only assign iroutes on global routes which span more than one complete global cell in a row or column. Local connections with pins and bends within global cells are delegated to the detailed router. This results in a formulation for the track assigner that can be solved quickly and provides large improvement on the run-time for the whole routing flow. A benefit of having an area router after track assignment is that some iroutes can be left unassigned. Any unassigned iroutes will be picked up and connected by the area router using the global route as a guide. Since iroutes within a row or column interact with each other, it is natural to formulate the track assignment problem on a whole row or column of global cells. Figure 1(A) depicts the global cells with horizontal and vertical tracks on the top row and left column of the global route cell layout. As input to the track assignment, we have

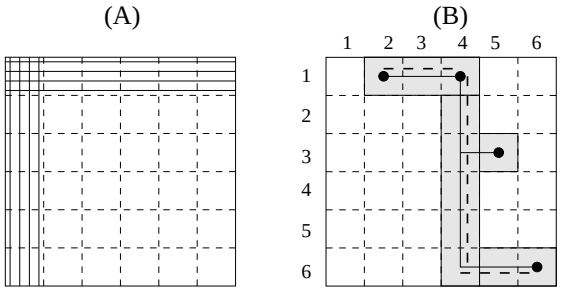

Fig. 1. Global cells and markings of horizontal and vertical grid lines in row and column respectively. Marking for a net shown in shaded GCs.

a partition of the routing area as a two dimensional array of global cells, and a global route for each net. The track assigner works on a full row or column of the GC array at a time. Each row (column) is called a *panel*. A panel has routing resources

in the form of gridlines (called *tracks*). Note that for a horizontal (vertical) panel, all tracks are horizontal (vertical). A track inside a horizontal (vertical) panel can be identified with its y (x) and z coordinate. The z-coordinate identifies the plane that the track lies on, and the y-coordinate (x-coordinate) gives its position on that plane. The length of horizontal (vertical) iroute is the absolute difference between the x-coordinates (y-coordinates). We exclude tracks that are not sufficiently spaced (at least line-to-line and preferably via-to-line apart) so that two iroutes can be placed on adjacent tracks without any spacing violations. A track can have obstructions marked onto it. These are caused by pre-routes, routing obstructions, macro-blocks etc. The global route for a net is typically represented by the set of GCs wherein the net's final physical connection has to be achieved. Figure 1(B) shows the GCs for a net with 4 netcs in shaded form. The global route of the net can also be visualized as a set of connections to be realized. In this example, these connections are $(1,2)$ to $(1,4)$, $(1,4)$ to $(6,4)$ to $(6,6)$ and $(3,4)$ to $(3,5)$. Note that these connections collectively cover all the GCs in the global route for the net. Note that the representation of a global route by a set of connections is not unique. Any representation works well for us, as long as it doesn't have unnecessarily small connections. We then break these connections into segments called *iroutes*. These iroutes are marked in solid dashed lines in Figure 1(B). Note that the small connection from $(3,4)$ to $(3,5)$ does not result in any iroute as it does not traverse a complete global cell. The three resulting iroutes belong to three different panels, two horizontal and one vertical. The coordinates of these iroutes determine the panels that contain them. The track assigner assigns iroutes in a panel to the available tracks in the same panel. It does so one panel at a time. For the rest of the paper, we will assume without loss of generality that the panels are horizontal. The mathematical formulation of the track assignment problem is as follows.

A. Problem Formulation

Let $\mathcal{T}$ be the set of tracks inside a panel. Let $\mathcal{I}$ be the set of iroutes which need to be track assigned in this panel. Each track $tr \in \mathcal{T}$ can be represented by its set of constituent contiguous intervals. Denoting these intervals by x_i, we have $tr \equiv \uplus x_i$. Each of this x_i is either

- a *blocked interval*, where no iroute from $\mathcal{I}$ can be assigned,
- an *occupied interval*, where an iroute from $\mathcal{I}$ has been assigned or
- a *free interval*, where no iroute from the set $\mathcal{I}$ has yet been assigned.

An iroute $ir \in \mathcal{I}$ is said to be assignable to $tr \in \mathcal{T}$, $tr \equiv \uplus x_i$, iff $x_i \cap ir \neq \emptyset$ implies that either x_i is a free interval or is an interval occupied by an iroute of the same net (as iroute ir). The track assignment problem can be defined as:

Track Assignment Problem: Given a set of tracks $\mathcal{T}$ and a set of iroutes $\mathcal{I}$, and a cost function $F : \mathcal{I} \times \mathcal{T} \rightarrow N$ which represents the cost of assigning an iroute to a track, find an assignment that minimizes the sum of the costs of the assignment.

B. Cost Metrics

We now describe the various factors that play a role in determining the cost of assigning an iroute ir to a track tr.

- **Plane cost:** In typical routing instances with standard cells, a large number of pins lie on layer zero. In order to connect to these pins, the resources on layer one must be carefully utilized. Placing long iroutes onto layer one (the layer immediately over the pins) hinders subsequent connections to pins on other nets (than the iroute). So it is desirable to put longer iroutes on higher layers. This is modeled through a *plane cost*. If tr is on plane p, and l is the length of the iroute ir, then this cost is proportional to $|l - p|$. This way longer iroutes are penalized if they are assigned to lower planes.
- **Track obstruction cost:** This component captures the assignability of an iroute ir to a track tr. If ir is assignable to tr, then this cost is zero. If this assignment is forbidden due to some blockages, pre-routes (say power or ground) etc. then its track obstruction cost is set to infinity. Note than once an iroute gets assigned to a track, it becomes an obstruction for iroutes belonging to other nets. The track obstruction cost for other iroutes overlapping onto this track becomes infinity.
- **Via obstruction cost:** An assigned iroute ir will connect with the rest of the net components and other assigned iroutes on the same net in the regions defined by the GCs. Sometimes global cells (in which connections have to be realized) are congested in layers above or below the track with pre-routes such as power or ground straps. In such a case, realizing a vertical connection will be impossible. The via obstruction cost prevents iroutes from being assigned to tracks that will be hard to connect.
- **Planar anchoring cost:** From the detailed routing perspective there are definitely preferred tracks for each iroute that help it to find a better solution. In the interest of minimizing jogs and wire length it is desirable to assign iroute ir as close as possible to other net components and assigned iroutes of its underlying net. For this, we would like to assign ir to a track tr whose y-coordinate minimizes the sum of the distances of connections to be made in the y direction. If y_t is the y-coordinate of the track tr and $Y_i \equiv \{y_1, y_2, \ldots, y_m\}$ is the set of y-coordinates of the gridlines where some net components and assigned iroutes of the same net are already marked, then this cost is proportional to $\sum_{y_i \in Y_i} |y_t - y_i|$.
- **Via anchoring cost:** In the interest of minimizing vias it is desirable to assign iroute ir on a plane as close as possible to other net components and assigned iroutes of its underlying net. This cost is similar to the planar anchoring cost, except that it is in the z-direction.

As a simple example consider the iroute in Figure 2. It is anchored to the pin on the right and the vertical assigned iroute A on layer four. Iroute A crosses tracks 3 and 4 on layer three. There is also an obstruction on track 4. For the horizontal iroute, track 1 has a cost of 3 (1 from the pin and 2 from the iroute A). Similarly tracks 2, 3 and 4 have a cost of 1, 1, and ∞ respectively. Note that some of these costs (or parts thereof)

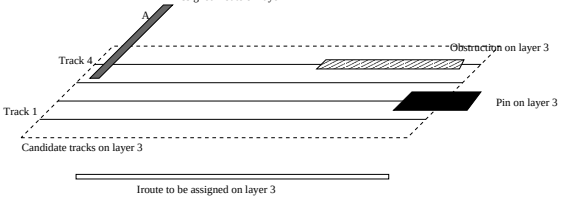

Fig. 2. Anchor costs and obstruction costs

are static, i.e. given $\mathcal{I}$ and $\mathcal{T}$ the costs can be calculated and stay unchanged. Example of such costs are plane costs. Also the components of track obstruction cost, via obstruction cost, planar anchoring cost and via anchoring cost due to net components are static. The dynamic portion of cost depends on

assignment of other iroutes. Components of the last four costs that rely on assigned iroutes belong to this category.

As mentioned in Section I, track assignment also lends itself as a suitable place to address several other DSM issues. Crosstalk avoidance, congestion minimization, process antennae issues etc. are to name a few of these. From the crosstalk perspective, one would like to put an iroute ir on a track tr whose neighboring tracks are relatively less populated. A cost metric capturing the presence of iroutes onto the neighboring tracks can easily be constructed. Preference to placing certain nets together (like placing a sensitive net near a ground strap for shielding) can also be modeled easily. Further, with the use of dummy obstacles (or dummy iroutes) extra spacing can be introduced wherever necessary. Similar measures can be taken to address the congestion issues. The issues related to process rules can also be addressed during track assignment. For example, the process antenna rule requires that there should be an upper bound on the metal being laid on a plane if the metal has a connection to a gate of a transistor. This translates to the the upper limit on lengths of the iroutes. If an iroute is of length longer than this upper limit, then the iroute can be split and it can be ensured that the split iroutes are laid on different planes. It is easy to construct cost metrics for other DSM issues as well. In our implementation we have considered the basic cost metrics mentioned in bullet points above.

Before concluding our discussion on cost metrics we would like to emphasize that there exists a strong interplay across these cost components. The overall cost of assigning ir to tr can be taken as a weighted sum of all these cost components. The weights need to be chosen with extreme care to prevent any extreme biasing towards one of the costs. The weights have to be chosen with appropriate consideration of various technology parameters and fine level implementation details. Further, the weights should be verified with extensive experiments over industrial benchmark circuits.

The track assignment problem has subtle nuances. With obstruction free tracks and without the cost metrics attached with each iroute-track assignment it can be easily solved using an $O(n.log\ n)$ algorithm [17]. However, with obstructions on tracks and the attached costs, even the feasibility part of the track assignment problem becomes intractable. It is NP complete to decide if all the iroutes in $\mathcal{I}$ are assignable to the available tracks in $\mathcal{T}$. The optimization problem of finding a minimum cost solution is harder than the feasibility problem and hence it is also NP complete. The proof for NP completeness follows from a reduction from Circular Arc Coloring Problem [18]. Since the details of the proof are not central to our discussion we defer this to Appendix A.

C. Underlying Graph Model

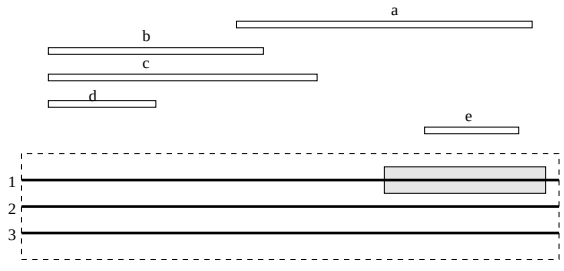

Fig. 3. Iroutes {a, b, c, d, e} and tracks {1, 2, 3} in the track assignment problem.

We now detail a graph model similar to the horizontal constraint graph model given in [19]. We add a weighted bipartite graph to model the iroute to track assignments. First we note that if the tracks are horizontal then each of the iroutes are also horizontal. Each iroute $ir \in \mathcal{I}$ can equivalently be represented by its underlying interval (l, u), where l is the smaller endpoint of the interval and u is the larger endpoint of the interval. The density of a point p on a track, denoted by d_p is equal to the number of iroutes in $\mathcal{I}$ whose corresponding interval contain this point. The density d of the set $\mathcal{I}$ is the maximum of d_p over all points p on the tracks. In order to have a feasible solution to the track assignment problem, we should have $d \leq |\mathcal{T}|$. We now build the graph model for the problem. We do this vis a vis an illustrative example. Let us consider the track assignment problem given in Figure 3. We have $\mathcal{I} = \{a, b, c, d, e\}$ and $\mathcal{T} = \{1, 2, 3\}$. Further, note that track 1 has an obstruction marked on it which prevents iroutes a and e to be assigned to it. We first build the *iroute overlap graph* $G(V_I, E_o)$. This is essentially the horizontal constraint graph. Each node $v \in V_I$ corresponds to an iroute in $\mathcal{I}$. Two nodes v_{i1} and v_{i2} are connected by an edge $e \in E_o$ iff these iroutes belong to two different nets and their spans overlap. An edge indicates that the two iroutes can not be assigned to the same track simultaneously. In such cases we call nodes v_{i1} and v_{i2} as *conflicting nodes* or *conflicting iroutes*. The graph $G(V_I, E_o)$ for the iroutes of example in Figure 3 is depicted in Figure 4. Let V_T denote the set of tracks in panel. Then the assignability of iroutes in V_I to the tracks in V_T is denoted by a bipartite graph $B(V_I, V_T, E_c)$, where E_c represents its edge set. The cost of an edge connecting iroute v_i to a track v_t is the cost of this assignment. The bipartite graph $B(V_I, V_T, E_c)$ for our illustrative example is also given in Figure 4. To be able to work in a unified model we merge these two graphs to get a unified graph model as illustrated in the same Figure. Note that the edges E_o of $G(V_I, E_o)$ are added to $B(V_I, V_T, E_c)$. Looking at this model, the track assignment problem reduces to matching nodes in V_I to the nodes in V_T, such that no two conflicting nodes in V_I are matched to same track in V_T. Note that multiple nodes from V_I can be matched to a single node in V_T as long as these nodes are not conflicting with each other. We are interested in minimum cost matching, amongst the ones which maximize the length of iroutes assigned to tracks. We have a polynomial algorithm for standard bipartite graph matching, but for this modified problem the problem is NP-complete. We hence solve the problem through a heuristic method based on weighted bipartite matching over the subproblems built from the main problem.

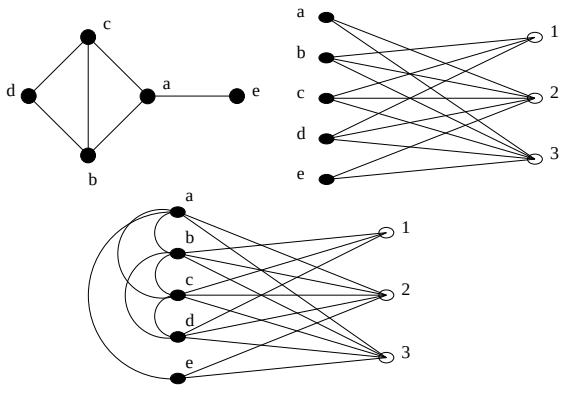

Fig. 4. Iroute overlap graph, bipartite assignment graph and the combined graph model.

D. Weighted Bipartite Matching

If we focus on one iroute, it is easy to sort the tracks according to its assignment costs. However, if two iroutes overlap each other they can not be assigned to the same track. Our algorithm selects a set of mutually conflicting iroutes and uses a weighted bipartite matching to assign them to different tracks. In order to do that, the algorithm first computes the maximal sets of conflicting iroutes. This is equivalent to finding the largest clique V_c in the graph $G(V_I, E_o)$. Since the graph is an interval graph it can be done in polynomial time [20]. The algorithm assigns one maximal subset of conflicting iroutes at a time starting from the largest subset. We model the problem as a minimum weighted bipartite matching problem and use the shortest augmenting path algorithm by Jonker and Volgenant [21] to solve it. After the assignment is done, the combined graph model is updated. The nodes in the clique are removed from the graph $G(V_I, E_o)$ and $B(V_I, V_T, E_c)$ along with their incident edges. If two conflicting iroutes v_i and v_j are such that, $v_i \in V_c$, $v_j \notin V_c$ and v_i is matched to $v_t \in V_T$, then we know that in v_j can never be assigned to v_t. Thus we remove the edge $\{v_i, v_t\}$ if it exists in E_c. After performing these modifications on the graph model, we determine once again the largest clique in the modified graphs and proceed as before. This heuristic will attempt to get a minimum weight matching for the entire problem. The heuristic however can easily miss certain obvious assignments. To further increase its performance we add look ahead into it.

E. Look-Ahead Heuristic

Let us reconsider our illustrative example. It may happen that the maximum clique detected is $\{b, c, d\}$. An assignment to tracks $\{3, 2, 1\}$ respectively prohibits assignment of a on any track. This solution is shown in Figure 5. We however know that a better solution, shown in Figure 6 does exist which can assign all iroutes. To do this we augment our graph update to incorporate a look ahead step. When we assign an iroute to a track we modify the graph model as explained above. If in this process, any iroute $v_i \in V_I$ becomes uniquely assignable to a track in V_T (i.e., there is only one edge from v_i to a node v_t in bipartite graph model), we accept this assignment immediately. We assign v_i to v_t and modify the graph model. In our example, assigning b to 3 makes a uniquely assignable to 2, hence we assign a to 2 and continue, and in the end we get the solution shown in Figure 6.

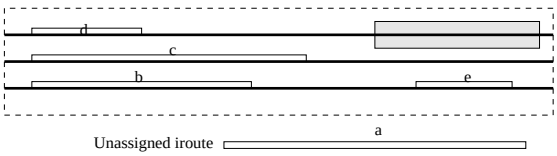

Fig. 5. Solution to the track assignment problem given in Figure 3.

Fig. 6. Solution to the track assignment problem given in Figure 3 with lookahead.

F. Iroute Splitting

At the end of the above mentioned clique based modified matching algorithm (with lookahead) it is possible that a few iroutes in $\mathcal{I}$ are still unassigned to any track in $\mathcal{T}$. These iroutes are unassignable to any tracks in $\mathcal{T}$. We have conducted experiments and found out that most of these iroutes span many GCs. In other words, the intervals corresponding to these iroutes are long. If left unassigned during track assignment the detailed router has to spend lot of effort realizing a physical connection in these long GC regions implied by these cells. It is worthwhile to consider assigning parts of these iroutes to tracks in $\mathcal{T}$. This reduces the search region for the detailed router. Let an iroute $ir \equiv (l, u)$ be one such unassigned iroute. We know that (l, u) is not assignable to any track in $\mathcal{T}$. We partition (l, u) into three intervals $(l, p_1), (p_1, p_2)$ and (p_2, u), where $l \leq p_1 < p_2 \leq u$, and the interval (p_1, p_2) is assignable to a track in $\mathcal{T}$. Looking into the blocked and occupied intervals on all the tracks, we find out the widest interval (p_1, p_2) which is assignable to at-least one track in $\mathcal{T}$. We then split the iroute ir into potentially three iroutes and assign the middle part to a track where it is assignable with least cost. We split and assign all the unassigned iroutes one by one.

G. Overall Track Assignment Algorithm

```
1   − Break global routes into segments
2   − Partition these segments into
3       horizontal and vertcal panels
4   − For each panel
5           − build the graph model
6               − build the segment overlap graph
7               − build the bipartite assignment graph
8           − determine the largest clique in segment overlap graph
9           − find the weighted bipartite matching for the segments
              in this clique to the tracks in the assignment graph
10          − for each of these segment to track assignment
11                  −perform the lookahead implications
12                  −update the graph model
13          − if assignable segments exist, goto step 8
14          − for each unassigned segment
15                  − find the track which can host its largest
                      part onto itself
16                  − split the segment around the largest part
17                  − assign the largest part onto the concerned track
18      − update the connectivity informations for assigned
          segments and netcs. Account for unassigned segments.
19      − proceed with detailed routing and ripup and re−routing
```

Fig. 7. Top level track assignment algorithm.

The track assignment stage is introduced in a standard routing flow as shown in Figure 7. After global routing, we detect the long segments from the global routes. We partition the GCs into panels, which are rows and columns of the GC array. We proceed on a per panel basis. We build the iroute overlap graph for the iroutes in a panel. The bipartite assignment graph is built by looking into the blockages for these iroutes onto the tracks. We then detect the largest clique in the iroute overlap graph and compute its minimum weight bipartite matching. Upon assignment of the matched iroutes to their tracks, we perform the lookahead implications. The graphs are updated with each assignment. We iterate this process on the modified graphs and continue till either all the iroutes are assigned or the unassigned iroutes have no feasible track. Unassigned iroutes are then split and we assign the split parts to the respective least cost tracks.

Finally, we update the connectivity information to account for all the iroute assignments and the failed assignments if any. The connectivity information can be thought of as regions composed of GCs to be used by the detailed router as a guide to find paths. The detailed router then looks for the connectivity information associated with each net (on a per net basis) and tries to determine a physical connection across all the constituent netcs and assigned iroutes of the net. After detailed routing, we go through a few passes of rip-up and re-route phases. In this step, a route for a net is removed if doing so facilitates routing of some other net(s) at lower cost. We would like to emphasize that the track assigned iroutes can also be ripped up during these passes. Though such rip-ups are performed at higher costs compared to metal laid by the detailed router. Thus the routing system can overcome poor decisions made by the track assignment step.

IV. Experimental Results

TABLE I
Specification of planes (p), cells, nets, pins and congestion (c) numbers for the test circuits

Circuit	p	cells	nets	pins	c
ckt1	5	7093	7223	37230	0.85
ckt2	5	89315	107711	343157	0.86
ckt3	5	34852	41751	202602	0.63
ckt4	5	5479	4311	26590	0.95
ckt5	3	13132	7598	48767	0.94
ckt6	5	90012	108466	344390	0.63
ckt7	3	38460	44369	130461	0.78
ckt8	4	106085	115986	555569	0.49
ckt9	4	176608	181822	573637	0.81
ckt10	5	13816	14032	73374	0.65
ckt11	3	40638	45862	148406	0.75
ckt12	5	11229	5390	41093	0.85
ckt13	5	24902	24131	135862	0.99
ckt14	6	149491	150211	500076	0.77

In this section, we support the suggested track assignment step by experimental observations. We have implemented the track assigner in our routing system. We report router performance with and without the use of track assignment stage. The experiments were conducted on several industrial circuits. The details of the circuits for which we present experimental results are reported in Table I. Note that circuits are taken from different technologies (3, 4, 5, and 6 layer), and the number of nets go up-to 180K. Most of these circuit have pre-routes, and blockages, which the router needs to respect. Congestion is defined as the ratio of total active cell area (discount filler cells, power decoupling capacitor cells and antenna diode cells) to the total layout area. Note that a few of these circuits are very congested designs. The weights for the various cost components are the same for all designs.

In Table II we compare the router performance in 2 stage and 3 stage routing. In 2 stage routing, global routing is followed by detailed routing, without any intermediate track assignment. In 3 stage routing, an intermediate track assignment stage is used. Table II contains the total run-time for the routing, peak memory usage, total number of vias used in the routing and the total length of the final routes which comes out after the detailed routing. It should be no surprise that the run-times for 3-stage routing are less compared to that of 2 stage routing. This is primarily due to the fact that the detailed router has less area to search in 3-stage routing. Also the time spent in track assignment during 3-stage routing is very small compared to the total runtime. However the 3-stage process comes with a minor excess use of routing resources (number of vias and length of routes). We conjecture that the via count loss can be easily recovered by a fast via-minimization step employed at the end. We get a runtime reduction by a factor or nearly upto 3, with extra routing resource of only 3-4%. All experiments were conducted on a 450MHz UltraSPARC-II machine with 4GB RAM. We conducted the experiments to measure the efficacy of various cost metrics. Just with the fundamental *via and plane obstruction costs* we get the runtime speedup as reported in Table II. With the use of *horizontal anchoring cost* the metal length on an average reduces by 2-3%. With the further use of *vertical anchoring cost* the number of vias also reduces by 4-5%.

Let L_{gr} be the total length of all the global routes put together. Let L_{ir} be the total length of all the iroutes. Note that an iroute was defined to be a segment in global route which spans at least two GCs. So only the highly zigzagged (or stair-cased) global routes escape from being broken into iroutes. For our test suite the ratio L_{ir}/L_{gr} is between 0.58 and 0.85. This ratio indicates the fraction of global routes which can potentially get track assigned. Further, let L^t_{ir} be the total length of iroutes which get track assigned by the track assigner. Then we can measure the success of the track assigner by the ratio L^t_{ir}/L_{ir}. We have found this ratio very close to 100% for most of the circuits in our suite. In Table III, we report the ratio L^t_{ir}/L_{ir} with and without the use of lookahead and segment splitting heuristics. It can be noted that with the use of heuristics the ratio improves substantially whenever it is far from 100%. The efficacy of the track assigner, in terms of reducing detailed routing efforts, can be measured by the product of L_{ir}/L_{gr} and L^t_{ir}/L_{ir}. Since L^t_{ir}/L_{ir} is very close to 100% for most of the designs, this product is very close to L_{ir}/L_{gr}. For all the circuits, the introduction of the track assignment stage did not affect the net completion. We report the percent iroutes (for each circuit) which are ripped up in Table III column (R). In most of the circuits less than 1% of the assigned iroutes are ripped up. For the most congested circuit (ckt7) it is 3.6%, which also is a small fraction.

The efficiency of our approach will be demonstrated by a comparison of a "back of the envelop" calculation to our method with the one proposed by Kay et. al. [15]. They use a Linear Programming relaxation of the Integer Program formulation of the track assignment problem. It is very easy to formulate the costs we discuss in section III-B in an ILP formulation. The constraints from crosstalk conflicts will not arise in our case (type III Clique constraints). On a 300MHz machine they assign 100 iroutes on 11 tracks (1100 variables and 1373 constraints) in 6 seconds and 530 iroutes on 23 tracks (12190 variables and 14457 constraints) in 15 seconds. Assuming that time complexity is linear in the constraint matrix size ($size$ = number of variables $\times$ number of constraints) we get $t \approx 5.113 \times 10^{-8} size + 6$. In Table III we report the number of panels for each design. Also given are average number of tracks in each panel and the average number or iroutes to be assigned on these tracks. Design 'ckt9' has 492 panels, with each panel having on an average 29 tracks and 142 iroutes. We have one constraint per iroute (type I Clique constraint) and one constraint per track for each clique in the interval graph (type II Clique constraint). We have on an average 100 cliques per panel leading to 4118 variables and 3042 constraints). This gives $t \approx 6.64$ seconds on a 300MHz machine. For the whole design it would take us 3267 seconds. On

TABLE II

COMPARISON OF RUNTIME (T), MEMORY REQUIREMENT (M), TOTAL VIA USED (V), TOTAL METAL LENGTH (L) USED FOR 2-STAGE (GR-DR CASE) AND 3-STAGE (GR-TA-DR CASE) ROUTING. IN 3-STAGE ROUTINE, TRACK ASSIGNMENT RUNTIME (TA-T) IS ALSO REPORTED.

	GR-DR				GR-TA-DR				
Circuit	T	M	V	L	TA-T	T	M	V	L
ckt1	238	48	56767	4.68977e+08	5	162	49	56787	4.72851e+08
ckt2	7286	570	867461	1.75286e+10	228	2806	607	897341	1.76642e+10
ckt3	1989	223	346632	5.66531e+09	60	949	231	353747	5.71506e+09
ckt4	137	23	41496	3.49537e+08	4	100	24	41800	3.52736e+08
ckt5	271	26	58252	8.58121e+07	13	197	27	57776	8.70529e+07
ckt6	6759	566	854603	1.72406e+10	214	2710	599	881910	1.73591e+10
ckt7	5986	147	366840	7.12562e+09	80	3100	168	376083	7.22332e+09
ckt8	8701	416	758855	2.13564e+09	284	3095	463	779783	2.14875e+09
ckt9	6656	524	1303603	2.03793e+10	151	3120	549	1298731	2.05398e+10
ckt10	363	20	111011	1.14971e+09	20	266	23	112447	1.15623e+09
ckt11	1180	44	352526	4.78811e+08	54	920	44	363546	4.86199e+08
ckt12	2446	40	46247	4.07718e+08	4	1845	40	45878	4.11807e+08
ckt13	862	129	214543	2.22958e+09	27	544	136	219549	2.24541e+09
ckt14	7104	752	1234374	1.0968e+10	205	3618	791	1275928	1.10529e+10

TABLE III

TRACK ASSIGNMENT RESULTS WITH (WI-H) AND WITHOUT (WO-H) THE LOOKAHEAD AND IROUTE SPLITTING HEURISTICS. ALL RATIOS ARE EXPRESSED AS PERCENTAGE. ALSO SHOWN ARE NUMBER OF PANELS (P), AVERAGE NUMBER OF TRACKS PER PANEL ($|\mathcal{T}|$), AVERAGE NUMBER OF IROUTES PER PANEL ($|\mathcal{I}|$) AND THE PERCENT OF ASSIGNED IROUTES RIPPED-UP (R) IN RIP-UP AND RE-ROUTE PHASE.

Circuit	P	$\|\mathcal{T}\|$	$\|\mathcal{I}\|$	R	L_{ir}/L_{gr}	L^t_{ir}/L_{ir}	
						WO-H	WI-H
ckt1	86	33	36	0.15	65.33	100.00	100.00
ckt2	627	34	134	0.11	85.12	99.89	99.99
ckt3	330	41	89	0.17	74.35	99.98	99.99
ckt4	68	37	39	0.41	62.94	100.00	100.00
ckt5	134	18	34	1.16	58.31	85.59	93.85
ckt6	627	34	124	0.09	84.87	99.98	100.00
ckt7	338	22	166	3.60	85.32	72.25	89.22
ckt8	704	29	159	0.19	80.98	98.89	99.72
ckt9	492	29	142	0.08	64.10	99.94	99.97
ckt10	144	33	51	0.10	60.22	100.00	100.00
ckt11	316	22	122	0.35	59.50	84.07	94.61
ckt12	78	29	39	3.38	64.48	99.93	99.93
ckt13	185	37	95	0.02	72.91	99.99	100.00
ckt14	513	40	152	0.04	75.02	99.97	100.00

a 450MHz machine this would take us $\approx$ 2156 seconds. This is approximately two-thirds the total routing time for our system. Using our approach we are able to perform track assignment for all these panels in 151 seconds (See column TA-T in Table II). This is a speedup of about 14 over the projected ILP based approach. Though we are unable to estimate the sub-optimality of the heuristic compared to the ILP, experimental results show that our approach is very effective in improving the run-time. Coupled with the facts that the L^t_{ir}/L_{ir} ratio is very close to 100% for most of the designs and a very small fraction of the iroutes are ripped, this leads us to conclude that our heuristic is able to solve the track assignment problem very efficiently.

V. CONCLUSIONS AND FUTURE WORK

We have suggested a track assignment stage between global routing and the detailed routing. This intermediate step reduces the run-times of overall routing process. The track assignment problem is posed as a modified bipartite graph matching problem. It is proved that the problem is NP complete and a heuristic based on weighted bipartite matching is suggested. Additional heuristics to improve the performance of the overall algorithm are also suggested. The combined cost metric is capable of addressing most of the routing constraints simultaneously. This includes efficient usage of routing resources. Finally, experiments on industrial circuits show the efficacy of the proposed approach.

The current implementation incorporates only the basic cost metrics which are aimed towards optimizing the usage of routing resources. We are currently working to incorporate other costs which address crosstalk, timing, congestion, antennae effect and other DSM problems. We are also investigating techniques to calculate gridlines efficiently in an environment where wire widths could differ on the same layer.

APPENDIX

I. NP HARDNESS OF TRACK ASSIGNMENT PROBLEM

In this section, we prove the NP completeness of feasibility solution to track assignment problem (TAP). We show that it is NP complete to decide if all the iroutes are assignable to available tracks in the presence of obstructions. We exhibit a polynomial time reduction of Circular Arc Coloring Problem (CACP) (which is known to be NP complete [18], [22]) to TAP. We use the notations given in [23] where the NP-completeness of Restricted Track Assignment Problem is proved using a reduction from CACP.

A circular arc graph is defined through a set of arcs $\mathcal{A}$ on a circle. If $|\mathcal{A}| = s$ then the arcs have $2s$ endpoints. Each arc $A_i \in \mathcal{A}$ can be thought of as a connection of its two endpoints (a_i, b_i) in a clockwise manner (See Figure 8(A)). Given any point P on the circle the subset $\mathcal{A}_P$ (where $\mathcal{A}_P \subseteq \mathcal{A}$) of arcs which contain this point are termed P-equivalent arcs. CACP seeks to assign colors to arcs in $\mathcal{A}$ such that every two P-equivalent

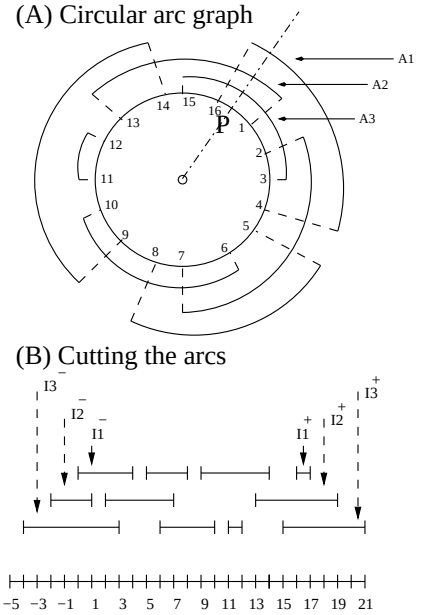

Fig. 8. Opening a circular arc representation.

arcs, for all points P on the circle, are assigned distinct colors. The maximum of $|\mathcal{A}_P|$, over all P, is called the density of $\mathcal{A}$ and is denoted by δ. It is NP-complete to decide if δ colors are sufficient to color an arbitrary instance $\mathcal{A}$ of CACP.

We now sketch the polynomial-time reduction of CACP to TAP. Consider an arbitrary instance $\mathcal{A}$ of CACP. Let P be a point with $\delta = |\mathcal{A}_P|$. Let $\mathcal{A}_P = \{A_1, A_2, \ldots, A_\delta\}$. The point P is selected such that it is not an endpoint of any arc in $\mathcal{A}_P$. See Figure 8(A) for an illustration of a circular arc graph. We seek to convert the circular arcs into linear intervals (equivalently iroutes) which can then be assigned to tracks (which are equivalent to colors in CACP). For this conversion we need to 'cut' the arcs at a suitably chosen point. The points on the circle are numbered 1 to n and the endpoints of the arcs in $\mathcal{A}$ lie on these points. We ensure that the point P lies in between n and 1; otherwise the points can be renumbered. We cut the arcs at the point P. The arcs $\mathcal{A}_P$ when 'cut' at point P result into intervals $\mathcal{A}_P^o$. The set $\mathcal{A}_P^o$ is obtained by cutting each arc $A_i = (a_i, b_i)$ in $\mathcal{A}_P$ into two intervals $I_i^- = (2-2i, b_i)$ and $I_i^+ = (a_i, 2s+2i-1)$ (See Figure 8(B)). The remaining arcs in $(\mathcal{A} - \mathcal{A}_P)$ can easily be though of as intervals. The complete set of intervals which need to be track assigned is $(\mathcal{A} - \mathcal{A}_P) \cup \mathcal{A}_P^o$. Let $\mathcal{I}_P = (\mathcal{A} - \mathcal{A}_P) \cup \mathcal{A}_P^o$. The density of iroutes $\mathcal{I}_P$ is clearly δ. We need to determine if $\mathcal{I}_P$ can be assigned onto tracks $\mathcal{T}$, such that $|\mathcal{T}| = \delta$. Such an assignment would result into a coloring in $\mathcal{A}$, provided that I_i^- and I_i^+ are assigned to same track for $1 \leq i \leq \delta$. Since the arcs in $\mathcal{A}_P$ need to be assigned δ distinct colors, we a-priori assign I_i^- and I_i^+ on track number i, for $1 \leq i \leq \delta$. Having assigned these iroutes we mark them as obstructions for the assignment of the rest of the iroutes in $(\mathcal{A} - \mathcal{A}_P)$.

Theorem A.1: An arbitrary instance of $\mathcal{A}$ is δ colorable if and only if all the iroutes in $(\mathcal{A} - \mathcal{A}_P)$ can be assigned on tracks in $\mathcal{T}$ (such that $|\mathcal{T}| = \delta$ and I_i^- and I_i^+ are marked as obstruction on track i, for $1 \leq i \leq \delta$).

Proof: (only if) If arcs in $\mathcal{A}$ are δ colored the iroutes are track assigned as follows. Assume A_i is assigned the color number i, for $1 \leq i \leq \delta$; otherwise, the colors are renamed. The intervals I_i^- and I_i^+ are already marked to track i. The rest of the intervals are assigned to the track number equal to the color number of the corresponding arcs.

(if) Consider an assignment of $(\mathcal{A} - \mathcal{A}_P)$ on $\mathcal{T}$. Since the iroute I_i^- and I_i^+ are marked as obstructions on track i, the corresponding arc A_i is hence assigned the color number i. For the remaining intervals, which have one to one correspondence with the arcs in $(\mathcal{A} - \mathcal{A}_P)$, the corresponding arc is assigned the color number same as the track number. A legal coloring of arcs in $\mathcal{A}$ is thus obtained. ■

TAP is in class NP, as we can non-deterministically assign the iroutes to tracks and check the validity of the assignment. Based on above theorem, we conclude that it is NP-complete to decide if the given set of iroutes are assignable to the available tracks in the presence of obstructions. To get a minimum cost assignment for TAP is also NP-complete.

References

[1] H. B. Bakoglu, *Circuits, interconnections, and packaging for VLSI*, Addison-Wesley, Reading, MA, USA, 1990.

[2] N. Sherwani, "Algorithms for VLSI physical design automation," *Kluwer Academic Press*, 1992.

[3] A. Hetzel, "A sequential detailed router for huge grid graphs," in *Design Automation Conference*, 1998, pp. 332–338.

[4] H.-P. Tseng, L. Scheffer, and C. Sechen, "Timing and crosstalk-driven area routing," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 20, no. 4, pp. 528–544, Apr. 2001.

[5] C.-C. Chang and J. Cong, "Pseudo pin assignment with crosstalk noise control," *International Symposium on Physical Design*, pp. 41–47, 2000.

[6] Y. S. Kuo, T. C. Chern, and Wei-Kuan Shih, "Fast algorithm for optimal layer assignment," *25th Design Automation Conference*, pp. 554–559, 1988.

[7] C.-J. R. Shi, "Solving constrained via minimization by compact linear programming," *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 635 –640, 1997.

[8] C.-C. Chang and J. Cong, "An efficient approach to multilayer layer assignment with an application to via minimization," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 18, no. 5, pp. 608–620, May 1999.

[9] M. Sriram and S. M. Kang, "Detailed layer assignment for MCM routing," *International Conference in Computer Aided Design*, pp. 386–389, 1992.

[10] J. M. Ho, M. Sarrafzadeh, G. Vijayan, and C. K. Wong, "Layer assignment for multichip modules," *IEEE Trans. CAD*, vol. 9, no. 12, pp. 1272–1277, December 1990.

[11] M. J. Ciesielski, "Layer assignment for VLSI interconnect delay minimization," *IEEE Trans. CAD*, vol. 8, no. 6, pp. 701–707, June 1989.

[12] P. Saxena and C. L. Liu, "Optimization of the maximum delay of global interconnects during layer assignment," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 20, no. 4, pp. 503–515, April 2001.

[13] J. D. Cho, S. Raje, M. Sarrafzadeh, M. Sriram, and S. M. Kang, "Crosstalk-minimum layer assignment," *Custom Integrated Circuits Conference*, pp. 29.7.1–29.7.4, 1993.

[14] J. Cong, J. Fang, and K. Y. Khoo, "DUNE: A multi-layer gridless routing system with wire planning," *International Symposium on Physical Design*, pp. 12–18, 2000.

[15] R. Kay and R. A. Rutenbar, "Wire packing: A strong formulation of crosstalk-aware chip-level track/layer assignment with an efficient integer programming solution," *International Symposium on Physical Design*, pp. 61–68, 2000.

[16] P. Groeneveld and L. P. P. P. van Ginneken, "Method of designing a constraint-driven integrated circuit layout," U.S. Patent number 6,230,304, May 2001.

[17] U. Gupta, D. T. Lee, and J. Leung, "An optimal solution for the channel assignment problem," *IEEE Transactions on Computers*, pp. 807–810, November 1979.

[18] M. R. Garey, D. S. Johnson, G. L. Miller, and C. H. Papadimitrou, "The complexity of coloring circular arcs and chords," *SIAM Journal on Algebraic Discrete Methods*, vol. 1, no. 20, pp. 216–227, june 1980.

[19] T. Yoshimura and E. S. Kuh, "Efficient algorithms for channel routing," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. CAD-1, no. 1, pp. 25–35, January 1982.

[20] F. Harary, *Graph Theory*, Addison-Wesley, Reading, MA, USA, 1972.

[21] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing 38*, pp. 325–340, 1987.

[22] M. R. Garey and D. S. Johnson, *Computers and Intractability A Guide to Theory of NP Completeness*, W.H. Fredman and Company, New York, 1979.

[23] M. Sarrafzadeh and D. T. Lee, "Restricted track assignment with applications," *International Journal of Computational Geometry and Applications*, vol. 4, no. 1, pp. 53–68, 1994.