An Auction Based Pre-Processing Technique to Determine Detour in Global Routing

Yue Xu and Chris Chu
Department of Electrical and Computer Engineering
Iowa State University, Ames, Iowa 50011-3060, USA
{yuexu,cnchu}@iastate.edu

***Abstract*—Global Routing has been a traditional EDA problem. It has congestion elimination as the first and foremost priority. Despite of the recent development for popular rip-up and reroute framework, the congestion elimination process remains arbitrary and requires significant tunings. In order to achieve more consistent congestion elimination, we propose a new pre-processing framework for global routing. In the framework, we first identify the most congested global routing locations by an interval overflow lower bound technique. Then we use auction based detour algorithm to compute which nets and where to detour. The framework can be applied to any global router and would help them to achieve significant improvement in both solution quality and runtime.**

I. Introduction

Global Routing is one of the most traditional computer aided physical design problem. In global routing, the layout regions are partitioned into coarse tiles. Capacity is assigned to grid edge abstracted from the routing resources between two neighboring tiles to limit the number of crossing wires. The fundamental goal for global routers is to realize all connections while minimizing total wirelength and conforming to all capacity constraints.

When wiring demands exceed resources, routing congestion arises. Congestion is an exacerbating issue for global routing due to the fact that the growth of wiring demands keeps on outpacing the growth of wiring resources [1]. The fact that the number of metal layers continues to grow is an evidence of routing resource shortage. Although we can always add more metal layers in theory, it is unwise to do so due to manufacturability and cost. To make the matter worse, modern IC designs tend to have extremely congested local regions caused by complicated on-chip communication, IP blocking or timing requirements. Global routers spend the majority of runtime to detour nets involved in congested regions. Detour usually comes along with a price tag of longer wirelength. It may significantly deteriorate the timing for a design and complicate detailed routing task. Congestion elimination becomes the key feature to tell the performance of global routers due to the increasing severity of congestion.

The most successful global routing framework for congestion elimination is the sequential rip-up and reroute framework [2]. In general, the sequential framework uses pattern routing [3] to initialize a routing solution. With the initial solution, the framework sequentially rips-up a single net currently using congested grid edges and reroutes the net to minimize its routing cost using maze routing [4] to. The greedy manner lacks the global view about the entire problem. In maze routing, the cost function plays a crucial role for routing quality. It needs to adaptively balance the extra wirelength and congestion reduction caused by detour. Unfortunately, a global router can never figure out whether to prioritize wirelength or congestion reduction for a particular net in an optimal way.

The two global routing contests sponsored by ISPD in 2007 [5] and 2008 [6] confirmed the leading position of sequential framework but also exposed its weakness. Top performers proposed during or after the contests include NTHU-R [7] [8], FGR [9], MaizeRouter [10], Archer [11], NTUgr [12] and FastRoute 3.0 [13]. They all adopt the rip-up and reroute framework and use history based cost function. In general, the cost function includes a historical term to increase the cost for consistently congested grid edges. Despite of the evolution of the cost function, global routers still requires significant tuning to function properly. The rip-up and reroute framework stays as a trial and error approach.

There exist a few concurrent global routing frameworks but their efficiency to eliminate congestion is questionable. The multicommodity flow based global router [14] showed inferior wirelength and congestion comparing to sequential routers even for easy benchmarks. For integer linear programming (ILP) based frameworks [15] [16] [17], the reluctance and difficulty to correctly detour is more obvious. Only after numerous efforts spent by ILP fail and indicate that current set of candidate routes cannot generate congestion free solutions, will concurrent routers include new candidates with more detours. Even though [17] showed significant improvement in terms of wirelength comparing to sequential routers, its excessive runtime prohibits it from practical usage. Besides, concurrent routers tend to generate results with more remaining congestions when the benchmarks are extremely hard.

In order to eliminate congestion in a more systematic manner, we propose an efficient pre-processing framework for global routing to simultaneously detour nets that interact with highly congested locations. The detour technique creates a detour edge for each net considered and mandates that routing for the net has to use the detour edge. In this way, we can effectively detour a net to reduce congestion while preserve the flexibility of routing in less congested region. The pre-processing framework draws lesson from the market mecha-

[1]This work was partially supported by IBM faculty award.

nism in economy theory. One of the fundamental principles of economy theory recognizes that a fully competitive market will optimally allocate resources and maximize social welfare. Our work designs an efficient market to let routing nets compete for precious routing resources to achieve better global routing solutions. The key contributions of this work include:

- A pre-processing framework to determine detour before global routing, which provides a new way to handle congestion elimination problem.
- An interval overflow lower bound technique that accurately computes the most congested interval for global routing. It provides a more realistic hot spot detection method for the pre-processing framework.
- An auction based detour algorithm that simultaneously determines detouring nets and locations, considering important factors like wirelength minimization and congestion reduction.

The flow of the detour pre-processing framework is shown in Fig. 1. Assume multi-pin nets have been broken down into 2-pin nets, we first identify the most congested interval. Then we use auction algorithm to simultaneously choose the nets to detour and calculate the detour locations in step 2. For each detoured 2-pin net $A \sim B$ with a detour edge $p_1 \sim p_2$, we will break the original 2-pin net into three parts: the detour edge $p_1 \sim p_2$ and two new nets $A \sim p_1$ and $p_2 \sim B$. The procedure is repeated until there exists no significant congestion. The modified 2-pin netlist is fed into a global router. The new global routing flow significantly improves the global routing wirelength and accelerates the convergence of global routing. The newly proposed pre-routing framework is compatible with any global router.

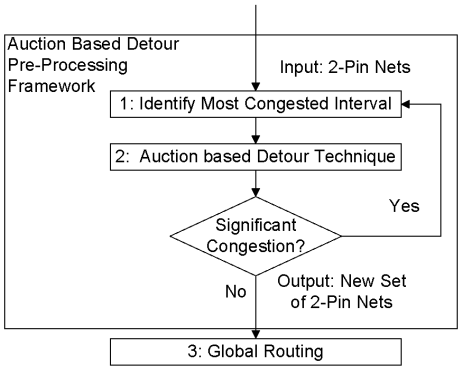
Auction Based Detour Pre-Processing Framework
Input: 2-Pin Nets
1: Identify Most Congested Interval
2: Auction based Detour Technique
Significant Congestion?
Yes
No
Output: New Set of 2-Pin Nets
3: Global Routing

Fig. 1. New Global Routing Flow

The rest of this paper is organized as follows. Sec. II presents the interval overflow lower bound technique while Sec. III details the auction based detour algorithm. Sec. IV provides the experimental results and the paper concludes with Sec. V.

II. Interval Overflow Lower Bound

In the detour pre-processing framework, we need to first identify the most congested locations.

One simple option for us is the currently available congestion estimation techniques [20] [21] [22]. We find using congestion estimator inaccurate for our purpose due to the different behavior between global routers and congestion estimators.

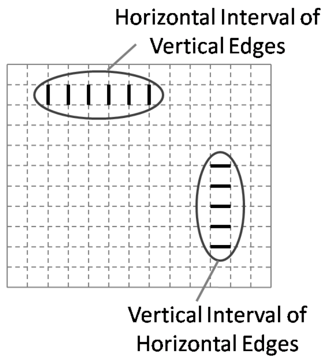
Horizontal Interval of Vertical Edges
Vertical Interval of Horizontal Edges

Fig. 2. Intervals of Grid Edges

We propose an interval overflow lower bound (IOLB) technique that calculates the lower bound of overflow on an interval. An interval is defined as a horizontal sequence of neighboring vertical grid edges or a vertical sequence of neighboring horizontal grid edges, as shown in Fig. 2.

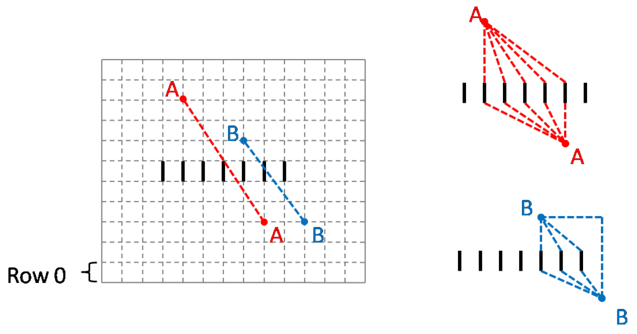

Fig. 3. Net-Interval Intersection

We call a net fully intersecting an interval if the bounding box of the net crosses the interval twice. In Fig. 3, net A fully intersects the interval but net B does not. Without detour, net A has to use one grid edge in the interval while net B has other choices. We define the demand D_I for an interval I as the total number of 2-pin nets that fully intersect the interval. D_I is the lower bound demand under the conditions of fixed topology and no detour. These two conditions generally hold for sequential global router when it generates tree structure and uses pattern routing to initialize the routing solutions. If the lower bound demand exceeds the total capacity C_I of the interval, we know for sure that some nets have to detour. So we deduct the capacities of the interval from the lower bound demand to generate interval overflow lower bound O_I, where $O_I = D_I - C_I$. We need to detour at least O_I nets that fully intersect I so that there would no longer be any congestion on the grid edges in the interval.

For a 2-pin net $(y_1, x_1) \sim (y_2, x_2)$, without loss of generality, we assume $y_1 < y_2$, $x_1 < x_2$ and denote $q = x_1 - x_2$. The 2-pin net has impacts on every horizontal interval between row y_1 and row $y_2 - 1$ that contains interval $x_1 \leftrightarrow x_2$. As shown in Fig. 3, row y consists of vertical grid edges between y and $y + 1$. If we go through every 2-pin net and update its demand on every interval it fully intersects, the run time would be $O(mn^3)$ where m is the number of nets and n is the maximum side size of global routing grid. This is too slow for large scale designs.

Algorithm: IOLB for Horizontal Intervals

```
1.  $\forall y, \forall x, \forall q, S_h[y][x][q] = 0$
2.  for every 2-pin connection $(x_1, y_1) \sim (x_2, y_2)$
3.      $x_{min} = min(x_1, x_2)$, $x_{max} = max(x_1, x_2)$
3.      $y_{min} = min(y_1, y_2)$, $y_{max} = max(y_1, y_2)$
4.      for $y = y_{min}; y < y_{max}; y++$
5.          $S_h[y][x_{min}][x_{max} - x_{min}] += 1$
6.  for $y = 0; y < yGrid - 1; y++$
7.      for $x = 0; x < xGrid; x++$
8.          $extra_h[x][0] = S_h[y][x][0] - cap_v[y][x]$
9.          for $q = 1; q < x; q++$
10.             $extra_h[x][q] = extra_h[x][q-1] + S_h[y][x-q][q]$
11.         $O_h[y][x][0] = S_h[y][x][0] - cap_v[y][x]$
12.     for $q = 1; q < xGrid; q++$
13.         for $x = 1; x < q - 1; x++$
14.             $O_h[y][x][q] = O_h[y][x][q-1] + extra_h[x+q][q]$
```

Fig. 4. Interval Overflow Lower Bound Algorithm for Horizontal Intervals

We observe that IOLB for an interval I_A consisting of $e_1, e_2, \ldots, e_q$ equals to the IOLB of the interval I_B consisting of $e_1, e_2, \ldots, e_{q-1}$, plus the number of nets that fully intersect I_A but not I_B, minus the capacity of the grid edge e_q. Based on the observation, we propose an algorithm based on dynamic programming that computes IOLB for all the intervals in $O(n^3)$ time. The algorithm to compute IOLB of a horizontal intervals is shown in Fig. 4.

In Fig. 4, $xGrid$ and $yGrid$ is the size of the global routing grid. y denotes the intervals on the y^{th} row of vertical grid edges, counting from the lower corner. x sets the left boundary index of the interval and q is the length of the interval. In the grid graph, a vertical edge is indexed according to its lower pin. $e_v[y][x]$ represents the grid edge between (y, x) and $(y+1, x)$ and $cap_v[y][x]$ is the capacity for $e_v[y][x]$. Horizontal grid edges are indexed according to their left pins similarly. $O_h[y][x][q]$ holds all the horizontal IOLB we want to compute.

In line 2 to 5, for each net, we add its demand to the shortest intervals it fully intersects from row y_{min} to row $y_{max} - 1$ to $S_h[y][x][q]$. Line 7 to 10 computes the number of nets that fully intersect interval $I[y][x][q]$ but not interval $I[y][x][q-1]$, minus the capacity of the edge $e_v[y][x+q]$. Line 11 initializes the IOLB for intervals consisting of a single grid edge. We use line 13 to add up the two parts of IOLB together for $I[y][x][q]$. The first part is the $O_h[y][x][q-1]$, IOLB for the interval one grid edge shorter. The second part is the pre-computed $extra_h[y][x][q]$. Thus, we can compute IOLB for all horizontal intervals based on efficient dynamic programming.

The three levels of for loops from line 6 to line 13 sets the complexity of the IOLB algorithm to $O(n^3)$. Because we generally have $O(n^3)$ intervals and IOLB computes all of them, there exists no superior algorithm with less complexity. We can obtain the interval overflow lower bound for vertical intervals similarly.

We denote the interval I^* with largest total overflow as the most congested interval because a global router has to detour O_{I^*} nets in order to eliminate congestion on the interval. So the larger O_{I^*} is, the more onerous task global router has.

IOLB technique with $O(n^3)$ complexity is not a trivial operation for large scale designs. Since the detour step that follows will change the routing solutions for nets fully intersecting the most congested interval, we need to update IOLB for impacted intervals. We use regional updating technique to confine the overflow calculation to intervals impacted by the detouring nets, instead of computing IOLB from scratch again. Although the regional update technique also has a complexity of $O(n^3)$, the size of the update region is much smaller than the entire global routing grids.

IOLB is the lower bound for congestion on an interval. Although it may under-predict the overflow significantly for intervals with few fully intersecting nets, it is tight for the most congested interval. Every net that does not fully intersect I would avoid using grid edges in the interval. Every net that fully intersect I has to use one grid edge in I if detour is banned. Thus we accurately counts the nets that have no choice other than the grid edges in I.

III. Auction Based Detour Algorithm

Once we find the most congested interval, we want to detour the smallest number of nets to eliminate the congestion on the interval.We view the problem as a competitions between 2-pin nets to stick to the grid edges in the interval so that their wire length will not be prolonged. This leads us to think about auction market. Auction market is considered very efficient in resource allocation and welfare maximization. If we can design a similar scheme for 2-pin nets to systematically bid for the limited number of routing resources in the interval and force nets with failed bid to use the grid edges outside the interval, we solve the congestion elimination problem for the most congested interval.

A. Detour Problem

The interval overflow lower bound calculated in Sec. II gives the least amount of nets that need to detour to eliminate congestion on the interval. All the detouring nets need to cross an extended interval of the original congested interval and the detour problem targets the issue that which nets should be detoured and where the detouring net should intersect the extended interval. The selection of detouring nets and the crossing locations needs to consider the impacts of congestion in affected region and the extra wirelength caused by detours.

For the most congested interval I_C, we extend the interval to find the shortest interval that fully contains I_C and can accommodate all crossing demands on itself. We denote the new interval as Extended Interval (EI). Fig. 5 shows an example of the extension process. For a given I_C, we will try to include one of the two grid edges neighboring the boundary of the congested interval. The grid edge that leads to an extended interval with smaller IOLB will be chosen. We call it EI evaluation. After one grid edge is included, if the extended interval has positive IOLB, we will extend EI to include the grid edge we do not choose in previous EI evaluation. In this way, we can achieve almost even extension

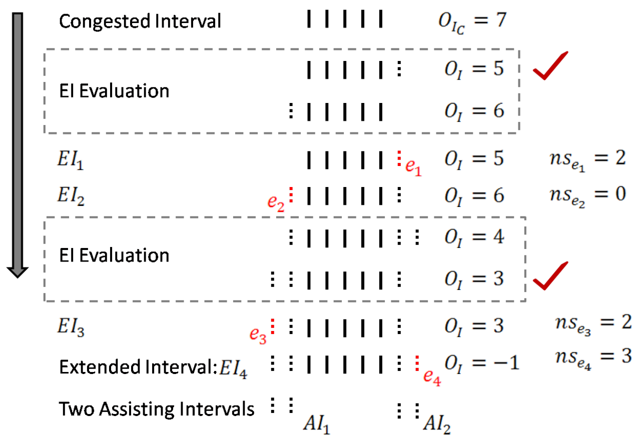

Fig. 5. Extension for Congested Interval

on both sides of I_C, which balances the detour on both sides and leads to shorter wirelength. If the new EI is still congested, we will go back to EI evaluation process. The EI evaluation and the possible inclusion of the forsaken grid edge is repeated until we find an interval with non-positive IOLB. For the above example, EI_4 is the final Extended Interval we want to find. The extended portions of EI is called assisting intervals, denoted as AI_1 and AI_2.

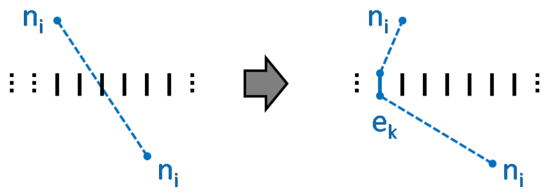

Fig. 6. 2-Pin Net Decomposition for a Determined Crossing

We create non-detour crossing sites on I_C and detour crossing sites on AI_1 and AI_2. Each crossing site corresponds to using one unit of routing usage. If one net n_i chooses one crossing site s_j on grid edge e_k, as shown in Fig. 6, its routing will consists of three components: the two smaller 2-pin sub nets and the usage of e_k. The pre-processing framework does not calculate the routing of 2-pin sub nets since we do not want to overly constraint the routing solution. Because the pre-processing framework focuses on detour that eliminates congestion, it only decomposes detouring nets to generate the new set of 2-pin nets but not non-detouring nets that use the crossing sites on I_C.

Algorithm: Detour Site Creation

```
1. $O_{min} = O_{I_C}$
2. $i = 1$
3. **while** $O_{min} > 0$
4.     $ns_{e_i} = max(0, O_{min} - max(O_{EI_i}, 0))$
5.     $O_{min} = min(O_{min}, O_{EI_i})$
6.     $i++$
```

Fig. 7. Detour Site Creation for Assisting Intervals

If the two assisting intervals consist of k grid edges, i.e. $e_1, e_2, \ldots, e_k$, indexed according to the sequence of their inclusion and the resulting extended interval after including grid edge e_i is labeled as EI_i, the number of detour crossing sites ns_{e_i} created for e_i is calculated in Fig. 7. The number of detour crossing sites created for e_i is basically $O_{EI_{i-1}} - O_{EI_i}$, the number of nets we can detour on e_i without inducing congestion on e_i. However, if we directly use $O_{EI_{i-1}}$ instead of O_{min} to calculate the number of sites, we would create 3 sites for e_3 and 8 sites in total for the example given in Fig. 5. 8 is larger than the IOLB of I_C because using $O_{EI_{i-1}}$ will create one detour crossing site to detour a net fully intersecting EI_2 but not I_C. Such detour should not be considered in the detour problem for I_C. Thus, we use O_{min} to guarantee that we create O_{I_C} detour crossing sites. In line 4, we have two max operations. The first max is used to prevent creating any detour crossing sites on an already congested e_i. The second max limits the detour crossing sites to O_{I_C} because the final EI may have a negative IOBL and lead to excessive detour crossing sites.

For each grid edge e_j in I_C, we create cap_{e_j} non-detour crossing sites. So in total, we create O_{I_C} detour crossing sites on assisting intervals and C_{I_C} non-detour crossing sites on I_C. For the detour problem, we only consider the D_{I_C} nets that fully intersect the most congested interval since we want to choose a subset of those nets form them to detour and determine their detour crossing sites on the assisting intervals. The number of crossing sites actually equals to the number of nets.

When crossing sites are created, each net and crossing site pair is assigned a weight which we give the name "quality loss", denoted as ql and defined in Equation 1:

$$ql_{i,j} = \alpha \cdot detour + \beta \cdot congestion \qquad (1)$$

In the above equation, $detour$ is the extra wirelength caused by the detour of net n_i using site s_j. $congestion$ models the impacts on congestion that would be caused by the two 2-pin sub nets after the net decomposition, as shown in Fig. 6. The congestion is calculated based on probabilistic Z routing presented in [22].

We generate a bipartite graph in which every net has a node residing on one side of the partition and every crossing site is given a node residing on the other side. There exists an edge between every pair of nodes on the different sides and the edge weight is set as the "quality loss".

We can use a minimum weighted bipartite matching to achieve the optimal set of crossing locations for all the nets. The minimum weighted bipartite matching solution corresponds to the optimal crossings in terms of wirelength and congestion impacts for the nets that cast their routing demands on I_C . However, the detour problem involve a dense graph with potentially millions of edges. Directly solving the bipartite matching would be very slow.

B. Problem Size Reduction

Since the detour problem size is very large, we use the following simplification techniques to identify which nets to detour and where they should detour more efficiently.

To reduce the problem size, we only create one crossing site for each grid edge on the extended interval, instead of multiple crossing sites on each grid edge. Each site can accept multiple crossings and the number of acceptable crossings is equal to the number of crossing sites we originally create for the grid edges.

The second type of simplification is congested interval abstraction. The nets assigned to use the original congested interval will not go through significant detour. Because the pre-processing framework focus on generating necessary detour to eliminate congestion, the non-detour crossings are purposefully ignored to maintain routing flexibility. The actual routing for non-detouring nets in the auction is left to global router to determine. Thus, it does not make much sense to compute where the non-detouring nets intersect with I_C. So we combine the grid edges in the original congested interval into a single crossing site, with the ability to accept C_{I_C} bids. The quality loss for this site is computed by assuming no detour and minimum congestion impacts.

In addition, we carry out a pre-selection procedure to limit the number of nets that would detour. It is obvious that some nets will use a lot of detour or cross significantly congested region to use any detour crossing site on the assisting intervals. They can be ignored in the auction without sacrificing the solution quality. So we selects $2O_{I_C}$ nets (if $2O_{I_C} < D_{I_C}$) with lowest "quality loss" to use the detour crossing sites. Meanwhile, the number of the non-detour crossing site is reduced accordingly to O_{I_C}.

The last type of speed up technique is side selection. The principle is very simple. Some of the $2O_{I_C}$ nets will obviously use one of the assisting interval instead of the other due to detour or congestion. During the pre-selection process, we can evaluate the quality losses derived from using the two assisting intervals. Comparing the two quality losses, we categorize the nets into three types: nets able to cross AI_1, nets able to cross AI_2 and nets able to cross both assisting intervals. The side selection limits the net to use detour crossing sites on one assisting interval and the non-detour crossing site if they are not categorized to be able to bid for both intervals.

The four types of simplification together can reduce the problem size by more than 95% and allow us to solve the detour problem for more congested intervals.

C. Auction Algorithm

Once we use the problem size reduction techniques, one crossing site can accept several nets to use it. Then we can no longer use the bipartite matching algorithm to solve the problem because it cannot handle multiple-to-one mapping. So we use auction algorithm to solve the detour problem because it can be tailored to solve the detour problem after problem size reduction.

We find an analogy between market economy and the detour problem. In the detour problem, every net tries to use smallest possible wirelength to complete routing and nets using the same congested grid edge are competing to stay still to preserve the shortest wirelength. If we treat every net as a bidder, every crossing site as an item and allow every net to seek after and compete for its desirable crossing sites, it would be like holding auctions to allocate routing resources of the extended interval. In fact,a few papers address the hidden link between auction algorithm and matching algorithm [23] [24].

The auction algorithm in [23] presents a two-phase framework to achieve an optimal assignment. In the first phase, every bidder calculates a bid for one item it wants the most. In the second phase, every item is assigned to its highest bidder. The two steps are repeated until every bidder gets an item.

Auction algorithms use four types of prices to guide the assignment. Firstly, every bidder has a fixed maximum bid it is willing to pay for each item to be auctioned. Secondly, every item has a price that always increases due to the bidding process in the auction. Thirdly, at the beginning of each round of auction, every bidder evaluates its benefit to buy each item depending on its maximum bid and current price for the item. An item with high maximum bid and low current price is very beneficial for a bidder because it may place a quite low bid to successfully buy the item. Finally, every bidder makes one bid for the item that brings maximum benefits to it.

For detour problem, the desirability for a net n_i to use the crossing sites on grid edge e_j is captured in the maximum bid $mb_{i,j}$ it is willing to make to use e_j. We use "quality loss" defined in the last section to calculate $mb_{i,j}$, the formula is shown below:

$$mb_{i,j} = (ql^* + 1) - ql_{i,j} \tag{2}$$

In Equation 2, $ql^* = max_{i,j}(ql_{i,j})$. We invert quality loss to achieve maximum bid because lower $ql_{i,j}$ indicates that it is more desirable for n_i to use e_j. We denote p_j as the price of e_j. The net benefit of an assignment Π (net to grid edge mapping) is defined as

$$\sum_{n_i}(mb_{i,\pi(i)} - p_{\pi(i)}) \tag{3}$$

We want to find Π^* that maximizes the net benefit. The auction algorithm is an iterative method to find the optimal prices and an assignment that maximizes the net benefits.

The auction algorithm in [23] cannot be directly used for the detour problem after problem size reduction because every item can only accept one bid. So we modify the second step of the auction algorithm so that every item, the crossing site on every grid edge, can accept up to ns_e bids. We create a priority queue of size ns_e for each edge to store temporary leading bidders. The modified auction algorithm is shown in Fig. 8.

In the modified auction algorithm, ϵ is introduced to prevent endless cycles. If the value of ϵ is chosen properly, the assignment generated by the terminated auction algorithm will satisfy "ϵ complementary slackness", i.e. all nets are assigned to grid edges that are within ϵ of being best [24].

Generally, if an item receives a bid in k iterations, its pries is at least $k\epsilon$. For a sufficiently large m, the item becomes very expensive and every nets still not assigned will avoid it. The auction algorithm converges in $O(mw^*/\epsilon)$ iterations, where m is the number of auctioned grid edge and w^* is defined as $max_{i,j}(w_{i,j})$. So the worst case complexity of the modified auction algorithm is thus $O(nm^2w^*/\epsilon)$, where n is the number of bidding nets. In practice, due to the variance of detour wirelength and congestion impacts, the increase in bidding price is much larger than ϵ. The auction algorithm runs quite efficiently.

Algorithm: Modified Auction Algorithm

```
1. Initialize the assignment $\Pi = \emptyset$, the set of unassigned
   net $I = n_1, \dots, n_n$, set prices $p_j = 0$ for all $e_j$.
2. Until $I$ is an empty set, the algorithm repeats the
   following two phases
3. Phase I: Bidding for all $n_i \in I$
4.     (1) Find benefit maximizing grid edge
          $j_i = argmax_j\{mb_{i,j} - p_j\}$
          $v_i = max_j\{mb_{i,j} - p_j\}$
          $u_i = max_{j \neq j_i}\{mb_{i,j} - p_j\}$
5.     (2) Compute the bid of bidder $n_i$ to grid edge $e_j$
          $b_{n_i \to e_j} = mb_{i,j_i} - u_i + \epsilon$
6. Phase II: Assignment for each grid edge $e_j$
7.     (3) Let $B(j)$ be the set of new bidders from which
          $e_j$ received a bid in the current iteration and $L(j)$
          be the set of nets leading the bid for and currently
          assigned to it. If $|B(j)| + |L(j)| > ns_{e_j}$, only
          the $ns_{e_j}$ highest bidder from $P(j) \cup L(j)$ will
          be assigned to $e_j$. If $|B(j)| + |L(j)| \leq ns_{e_j}$, all
          $n_i \in B(j)$ will be assigned to $e_j$. $p_j$ will be set
          to the lowest bidding price of the accepted net and
          the priority queue for temporary leading bidders
          will be updated.
```

Fig. 8. Modified Auction Algorithm

For the original auction algorithm, in which every item accepts only one bid, [25] proved that the total net benefits is within $n\epsilon$ of being optimal. The modified auction algorithm can be proved to be within $n\epsilon$ of being optimal in a similar manner and we skip the proof due to space limits.

The modified auction algorithm is a relaxation solution for the original detour problem and it has several advantages. It generates a solution within the vicinity of optimal solution in theory. Our experimental results support such theoretical solution quality. On the other hand, the modified auction algorithm can be fully parallelized. The bidding price evaluation of each net is independent from other nets and the assignment procedure for every grid edge is independent from other grid edges. So the auction algorithm is very suitable for today's multicore platforms. More importantly the auction algorithm provides us a method to simultaneously decide crossings and detour a minimum number of nets to eliminate the congestion over the interval with largest IOLB.

D. Solution Refinement

We use the assignment solution to modify the netlist of original benchmarks and feed the new set of 2-pin nets that contains detour decisions to the global router. One of our concerns is that the detour algorithm might hamper the flexibility of solutions or cause congestion in the neighboring region where detour crossing sites are created. To such problems, we design a solution refinement stage to use maze routing to reroute the 2-pin nets that are decomposed in the pre-processing stage. Such solution refinement brings in less than 0.1% wirelength improvement, which indicates that the original pre-processing framework is performing well. Thus, we removed the solution refinement due to its inefficacy.

IV. Experimental Results

We implement the pre-processing framework in C and use FastRoute 4.0 [26] as the following global router. All the experiments are conducted on a Linux machine with a $2.6GHZ$ Intel Processor and $32GB$ memory. We compare the results of our work with the winning global routers in ISPD 08 global routing contests and the benchmarks we use are from ISPD 08 contest too.

We separate the benchmarks into two categories: routable benchmarks and unroutable benchmarks. The performance comparison for routable benchmarks is shown in Table I. Comparing to the contest winners, our work achieves shortest wirelength with much less run time. Our work achieves 2.5%, 1.1% and 4.9% less wirelength comparing to FastRoute 4.0, NTHU-R 2.0 and NTUgr respectively. In addition, the new routing framework runs $1.1X$, $1.4X$ and $19.8X$ faster than the three global routers.

The performance comparison for unroutable benchmarks is shown in Table II. Again, our work uses least amount of time to generate solutions with shortest wirelength. It is worth noticing that our work also achieves the smallest number of remaining overflow for 3 benchmarks, which demonstrates the effectiveness of the pre-processing framework that simultaneously determines detour. For "newblue3", the solution of our work falls short by 104 nets or 0.3% from the best result generated by NTUgr. However, our work only spends 1.7% runtime of NTUgr for this specific benchmark.

Table III shows the run time decomposition of our framework. For easy benchmarks, it spends a significant portion of runtime to calculate IOBL and host auctions while saving relatively little runtime from global router. As benchmarks become larger and harder, the advantage starts showing up. For hard benchmarks, IOBL and auction algorithm spend less than 20% of total runtime. Because the framework actually outruns FastRoute 4.0 by another $20\% \sim 40\%$ for hard benchmarks, the pre-processing framework effectively reduces the runtime of the following global routing stage by around 50%.

TABLE III
Runtime Decomposition

Name	IOBL	Auction	Global Routing
adaptec1	1%	45%	54%
adaptec2	7%	22%	71%
adaptec3	7%	40%	53%
adaptec4	21%	8%	70%
adaptec5	2%	24%	74%
bigblue1	1%	17%	82%
bigblue2	3%	19%	78%
bigblue3	7%	24%	68%
bigblue4	1%	19%	80%
newblue1	2%	13%	85%
newblue2	9%	9%	82%
newblue3	1%	9%	90%
newblue4	1%	12%	87%
newblue5	5%	14%	81%
newblue6	3%	16%	81%
newblue7	0.2%	5%	94.8%

V. Conclusions

In this work, we propose a detour pre-processing framework for global routing. The framework consists of a two step

TABLE I
COMPARISON OF ROUTING RESULTS ON ROUTABLE BENCHMARKS

	Our Work		FastRoute 4.0 [26]		NTHU-R 2.0 [8]		NTUgr [12]	
Name	Wirelength	cpu(s)	Wirelength	cpu(s)	Wirelength	cpu(s)	Wirelength	cpu(s)
adaptec1	5352K	400	5460K	279	5344K	611	5740K	291
adaptec2	5151K	89	5277K	59	5229K	102	5370K	71
adaptec3	12922K	384	13213K	240	13101K	549	13500K	284
adaptec4	12021K	116	12249K	41	12169K	130	12370K	78
adaptec5	15394K	643	15866K	660	15538K	1160	15990K	988
bigblue1	5598K	333	5775K	530	5595K	690	6000K	1169
bigblue2	8919K	462	9352K	792	9059K	427	9120K	16044
bigblue3	12899K	203	13073K	158	13068K	253	13350K	258
newblue1	4575K	431	4686K	377	4653K	312	4930K	63161
newblue2	7459K	70	7636K	18	7570K	61	7690K	39
newblue5	22909K	595	23377K	777	23158K	977	24490K	1324
newblue6	17560K	565	18078K	884	17689K	912	18660K	1376
Comparison	1	1	1.025	1.12	1.011	1.44	1.049	19.8

TABLE II
COMPARISON OF ROUTING RESULTS ON UNROUTABLE BENCHMARKS

	Our Work			FastRoute 4.0 [26]			NTHU-R 2.0 [8]			NTUgr [12]		
Name	Overflow	WL	cpu(s)	Overflow	WL	cpu(s)	Overflow	WL	cpu(s)	Overflow	WL	cpu(s)
bigblue4	132	22887K	1431	150	25147K	3640	162	23090K	6633	188	24280K	26692
newblue3	31128	10615K	1749	31634	10752K	1181	31454	10653K	6168	31024	18830K	57120
newblue4	130	12964K	1208	140	13821K	2382	138	13046K	4873	142	14380K	72246
newblue7	54	35177K	7312	58	35974K	10209	62	35522K	7252	310	37220K	93401
Comparison	1	1	1	1.017	1.050	1.49	1.012	1.008	2.13	1.007	1.16	21.3

flow . It first accurately identifies the most congested interval. Later, it uses modified auction algorithm to simultaneously detour a minimum number of nets fully intersecting the interval to eliminate congestion. The pre-processing framework significantly improves the performance of FastRoute 4.0. The results outperform state-of-the-art global routers in wirelength, overflow and runtime.

REFERENCES

[1] International Technology Roadmap for Semiconductors 2009 Edition, http://www.itrs.net/

[2] S. M. Sait and H. Youssef, "VLSI Physical Design Automation," World Scientific, 1999.

[3] R. Kastner, E. Bozorgzadeh and M. Sarrafzadeh, "Pattern Routing: Use and Theory for Increasing Predictability and Avoiding Coupling," *Proc. of IEEE transactions on Computer-Aided Design of Integrated Circuits and Systems*, VOL. 21, NO. 7, 2002.

[4] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numerische Mathematik 1, pp. 269-271, 1959.

[5] http://www.sigda.org/ispd2007/rcontest/.

[6] http:http://www.ispd.cc/contests/ispd08rc.html.

[7] J.-R. Gao, P.-C. Wu, and T.-C. Wang, "A new global router for modern designs," *Proc. Asia and South Pacific Design Automation Conf.*, pp. 232-237, 2008.

[8] Y.-J. Chang, Y.-T. Lee, and T.-C. Wang, "NTHU-Route 2.0: A Fast and Stable Global Router," *Proc. IEEE/ACM Intl. Conf. on Compuer-Aided Design*, pp. 338-343, 2008.

[9] J. A.Roy and I. L. Markov, "High-performance routing at the nanometer scale," *Proc. IEEE/ACM Intl. Conf. on Compuer-Aided Design*, pp. 496-502, 2007.

[10] M. D. Moffitt, "MaizeRouter: Engineering an effective global router," *Proc. Asia and South Pacific Design Automation Conf.*, 2008.

[11] M. M. Ozdal and M. D.F. Wong, "ARCHER: a history-driven global routing algorithm," *Proc. of Intl. Conf. on Compuer-Aided Design*, pp. 481-487, 2007.

[12] Y. Chen, C. Hsu and Y. Chang"High-Performance Global Routing with Fast Overflow Reduction," *Proc. Asia and South Pacific Design Automation Conf.*, pp. 582-587, 2009.

[13] Y. Zhang, Y. Xu and C. Chu, "FastRoute3.0: a fast and high quality global router based on virtual capacity," *Proc. of Intl. Conf. on Compuer-Aided Design*, pp. 344-349, 2008.

[14] C. Albrecht, "Provably good global routing by a new approximation algorithm for multicommodity flow", *Proc. of Int'l Symp. on Physical Design*, pp. 19-25, 2000.

[15] M. Cho and D. Z. Pan, "BoxRouter: A new global router based on box expansion and progressive ILP," *Proc. of Design Automation Conference*, pp. 373-378, 2006.

[16] M. Cho, K. Lu, K. Yuan and D. Z. Pan, "BoxRouter 2.0: Architecture and Implementation of a hybrid and robust global router," *Proc. of Intl. Conf. on Compuer-Aided Design*, pp. 503-508, 2007.

[17] T. Wu, A. Davoodi and J. Linderoth, "GRIP: scalable 3D global routing using integer programming," *Proc. of Design Automation Conference*, pp. 320-325, 2009.

[18] R.T. Hadsell and P.H. Madden, "Improved global routing through congestion estimation," *Proc. of Design Automation Conference*, pp. 28-31, 2003.

[19] M. Pan and C. Chu, "FastRoute: A step to integrate global routing into placement," *Proc. of Intl. Conf. on Computer-Aided Design*, pp. 464-471, 2006.

[20] J. Lou, S. Krishnamoorthy and H. Sheng, "Estimating routing congestion using probablistic analysis," Proc. Intl. Symp. On Physical Design, pp. 112-117, 2001.

[21] A. B. Kahng and X. Xu, "Accurate pseudo-constructive wirelength and congestion estimation," *Proc. Intl. Workshop on System-Levvel Interconnect Prediction,* pp. 61-68, 2003.

[22] J. Westra, C. Bartels and P. Groeneveld, "Probablistic Congestion Prediction," *Proc. Int. Symp. on Physical Design*, pp. 204-209, 2004.

[23] M. Bayati, D. Shah and M. Sharma, "A Simpler Max-Product Maximum Weight Matching Algorithm and the Auction Algorithm," *IEEE transactions on Information Theory*, VOL. 54, NO. 3, pp. 1241-1251, 2008.

[24] D. Bertsekas, "Auction algorithms for network flow problems: A tutorial introduction," *Computational Optimization and Applications*, VOL. 1, NO. 1, pp. 7-66, 1992.

[25] D. Bertsekas and J. Tsitsiklis, "Parallel and Distributed Computation: Numerical Methods," Prentice-Hall, Englewood Cliffs, J.J., 1989.

[26] Y. Xu, Y. Zhang and C. Chu, "FastRoute 4.0: global router with efficient via minimization," *Proc. Asia and South Pacific Design Automation Conf.*, pp. 576-581, 2009.