A Game Theory-Based Heuristic for the Two-Dimensional VLSI Global Routing Problem

Umair F. Siddiqi
Center for Communications & Information Technology Research, Research Institute, King Faisal University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia
ufarooq@kfupm.edu.sa

Sadiq M. Sait†
Center for Communications & Information Technology Research, Department of Computer Engineering, College of Computer Science & Engineering, King Faisal University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia
sadiq@kfupm.edu.sa

Yoichi Shiraishi
Department of Mechanical Science and Technology, School of Science and Technology, Gunma University, 29-1 Honcho, Ota-Shi, Gunma 373-0052, Japan
yoichi.sirai@gunma-u.ac.jp

Received 10 June 2014
Accepted 9 February 2015
Published 19 March 2015

In this work we propose a game theory (GT)-based global router. It works in two steps: (i) Initial routing of all nets using maze routing with framing (MRF) and (ii) GT-based rip-up and reroute (R&R) process. In initial routing, the nets are divided into several small subsets which are routed concurrently using multithreading (MT). The main task of the GT-based R&R process is to eliminate congestion. Nets are considered as players and each player employs two pure strategies: (attempt to improve its spanning tree, and, do not attempt to improve its spanning tree). The nets also have mixed strategies whose values act as probabilities for them to select any particular pure strategy. The nets which select their first strategy will go through the R&R operation. We also propose an algorithm which computes the mixed strategies of nets. The advantage of using GT to select nets is that it reduces the number of nets and the number of iterations in the R&R process. The performance of the proposed global router was evaluated on ISPD’98 benchmarks and compared with two recent global routers, namely, Box Router 2.0 (configured for speed) and Side-winder. The results show that the proposed global router with

*This paper was recommended by Regional Editor Emre Salman.
†Corresponding author.
MT has a shorter runtime to converge to a valid solution than that of Box Router 2.0. It also outperforms Side-winder in terms of routability. The experimental results demonstrated that GT is a valuable technique in reducing the runtime of global routers.

Keywords: Global routing; game theory; rip-up and rerouting; VLSI physical design.

1. Introduction

Global routing in a critical step in the physical design of integrated circuits and is an NP-hard problem. It lies between the placement and detailed-routing steps in the physical design of VLSI chips. In global routing, the nets of wires are mapped to a coarse grid of global routing cells (or gcells). Each gcell has a fixed horizontal and vertical capacity. The task of global routing is to assign the nets while satisfying the capacity constraints of gcells. Each net is routed by generating a spanning tree for it that covers all of its pins. A solution of the global routing problem which does not violate the capacity constraints of gcells is called a valid solution. Routability-driven (RD) placement is a recent development that uses global routing to guide the placement process. The RD placement process will need to execute global routing several times. Therefore, the global router should not only be fast but must also yield good results.

Among the methods of routing, maze routing is the only method that guarantees to find a path between any two pins if there exists one. Therefore, many global routers use maze routing exclusively, or use other methods for initial routing and use maze routing for difficult-to-route nets. However, maze routing is slow and memory intensive. Maze routing is generally used with a rip-up and reroute (R&R) process to produce valid solutions. The main task of the R&R process is to selectively R&R a small fraction of nets in order to eliminate congestion. Maze routing with framing (MRF) is a modification of maze routing in which a net determines its spanning tree within a bounding box on the grid. The MRF method is very fast as compared to traditional maze routing and the size of the bounding box can be increased or decreased. Lee algorithm is a popular method of maze routing and has a breath-first behavior. Recent research showed that Lee algorithm is highly suitable for implementation using parallel computing platforms such as graphics processor units (GPUs) because of its simple data-structure and breath-first behavior which can be easily implemented on parallel platforms. A review of recently proposed global routers can be found in Ref. 11.

This work proposes a simple global router that uses MRF method exclusively. It also contains a R&R process whose role is to eliminate congestion. The R&R process is modeled using game theory (GT) in which nets act as players that want to optimize their spanning trees in terms of congestion and wire-length. GT is a very useful technique of decision making and multi-agent optimization. In the literature, GT has been successfully used in routing problem of communication networks and multi-agent optimization problem. GT is considered a viable tool in
wireless networking to solve routing issues.\textsuperscript{16} However, it is the first time that GT is applied to the global routing problem.

The performance of the proposed global router was evaluated on industrial benchmarks ISPD'98.\textsuperscript{17}\textsuperscript{a} The proposed global router is compared with two recent integer linear programming (ILP)-based global routers because they share a common property of employing optimization to eliminate congestion. The selected global routers are: (i) Box Router 2.0\textsuperscript{18,19} whose parameters were configured for speed and (ii) Side-winder.\textsuperscript{20,21} Box Router 2.0 has won third place in the ISPD 2007 global routing contest.\textsuperscript{19} It employs a method of routing in which a smaller region (referred as box) is selected on the routing grid and nets that lie within the box are routed and congestion is minimized (or dispersed) using ILP. The box is initiated on the most congested region of the routing grid and gradually expands to cover the entire routing grid. It employs pattern routing and adaptive maze routing to route the nets. Side-winder is another recent global router in which the ILP technique is global and covers the whole grid.

The experimental results show that the proposed global router can solve all problems of ISPD’98 suite. It is fast as well as it can produce smaller wire-length. When compared with the speed optimized version of Box Router 2.0,\textsuperscript{18,19} it has better runtime and wire-length. When compared with Side-winder,\textsuperscript{20,21} it has better routability because Side-winder did not solved all problems of the ISPD’98 suite.

This paper is organized as follows. Section 2 describes the global routing problem. Section 3 describes the design of the proposed global router and its GT-based heuristic. Section 4 discusses the experimental results. Section 5 contains the conclusion.

2. Global Routing Problem

The global routing problem is modeled using a two-dimensional (2D) grid-graph $G$. The grid-graph has a set of vertices $V$ and a set of edges $E$. Each vertex $v_i \in V$ corresponds to a gcell, and each edge $e_{ij} \in E$ corresponds to a boundary between adjacent vertices $v_i$ and $v_j$. Each edge $e_{ij}$ has a capacity $c_{ij}$ which is the maximum number of nets or wires that can pass through it. The actual number of nets that are passing through an edge $e_{ij}$ is called its demand and is represented as $u_{ij}$. The problem also contains a set of nets $N$, where each net $n_i \in N$ is composed of a set $P_i$ of pins (with each pin corresponding to a vertex $v_i$). Each net $n_i \in N$ is routed by finding a tree $t_i$ that covers all its pins. Each tree $t_i \in E$ is a set of edges without any cycles or repetition of edges and $t_i \subseteq E$. The set $T$ stores the spanning trees of all nets.

The primary objective of global routing is to route all the nets and to make sure that the capacity constraints of the edges are not violated, i.e., $u_{ij} \leq c_{ij}, \forall e_{ij} \in E$. The edge $e_{ij}$ is said to be congested or have an overflow if $u_{ij} > c_{ij}$. The edges whose

\textsuperscript{a}http://cseweb.ucsd.edu/~kastner/labyrinth_vault/benchmarks/index.html (accessed on 30 December, 2014).
demand is equal to their capacity are said to be fully utilized. The fully utilized and congested edges are considered important in eliminating congestion. The congested edges contribute directly to the total overflow (tof). The fully utilized edges although do not contribute directly to the congestion, but they can contribute to blocking routing of the other nets. For any edge \( e_{ij} \), the amount of overflow on it i.e., \( \text{overflow}(e_{ij}) \), and to determine if it is fully utilized i.e., \( \text{full}(e_{ij}) \) can be expressed using the following equations:

\[
\text{overflow}(e_{ij}) = \begin{cases} u_{ij} - c_{ij} & \text{if } u_{ij} > c_{ij} \\ 0 & \text{otherwise,} \end{cases}
\]

\[
\text{full}(e_{ij}) = \begin{cases} 1 & \text{if } u_{ij} = c_{ij} \\ 0 & \text{otherwise.} \end{cases}
\]

The tof is defined as equal to the tof of all edges and can be computed as follows:

\[
\text{tof}(T) = \sum_{e_{ij} \in E} \text{overflow}(e_{ij}).
\]

3. Proposed Global Router

3.1. Overview

The proposed global router executes three main tasks: (a) ordering of nets (b) initial routing of nets and (c) R&R process to eliminate congestion from the solution of initial routing. The tasks are distributed among several components. Figure 1 shows the UML component diagram\(^b\) of the proposed global router that shows that it has three main components: (i) Initial Routing, (ii) MRF method and (iii) R&R process. These components work together to accomplish global routing. The component Initial routing executes the tasks of ordering and routing the nets. It uses the component MRF method to route the nets. The work of the R&R process component

\(^b\)http://www.uml.org/ (accessed on 30 December, 2014).
starts after the completion of the initial routing phase and has several sub-components. The sub-component Selection of nets for the R&R operation uses the component GT-based heuristic to select nets. The component GT-based heuristic applies GT to determine the nets that should be ripped-up and rerouted in order to reduce the congestion of the solution. The nets selected for the R&R operation will be stored in the set $SS$ and sent to the component Apply R&R operation. The component Update parameter values performs self-adjustment of parameters values based on the feedback of previous iterations in order to reduce the convergence time of the R&R process.

Figure 2 shows the different steps in the proposed global router. The first step is the reading of the 2D grid graph $(G(V, E))$, set of all nets $(N)$ and parameters. The complete list of all parameters is described in Sec. 3.3.3, however, one parameter is described here which is $NUM\_THREADS$. $NUM\_THREADS$ sets the number of concurrent threads in the initial routing of nets. The second step is the sorting and initial routing of the nets. The nets are sorted in an ascending order of the area bounded by their pins. In this ordering, the nets whose pins are closer to each other are routed first. The rationale behind this ordering is that the nets whose pins are far from each other usually have more number of possible trees (i.e., alternate choices) available to them as compared to the nets whose pins are closer to each other. The nets that have more alternate choices available are less likely to be blocked from the routing of the other nets. The routing of nets can be done sequentially or concurrently. Most of the processors that are available these days have multi-cores and are capable of multithreading (MT). The routing of nets can be parallelized by using the following steps: (i) dividing the sorted nets in $N$ among $\{N_0, N_1, \ldots, NUM\_THREADS-1\}$ subsets, (i) creating concurrent threads $tr_0, tr_1, \ldots, tr_{NUM\_THREADS-1}$ such that $tr_i$ should route the nets in $N_i$ (where $i = 0$ to $NUM\_THREADS$) and (iii) executing the threads in parallel. The threads should update the usage information of the edges after routing each net. The parallelization causes some increase in the congestion of the initial routing phase but that can be

![Fig. 2. Overall flow of the proposed global router.](image)
compensated in the GT-based R&R process. After the completion of the initial routing phase, the R&R process is executed until the tof becomes zero.

### 3.2. MRF method

In MRF method, the routing region of each net is restricted to a rectangular region that covers its pins and some surrounding cells. Figure 3 shows the bounding region of a net which has three pins \((p_0, p_1, p_2)\). The size of the bounding box can be increased and decreased using the parameter \(BOX\_SIZE\). Figure 4 shows the method of routing the nets. The variable \(t_i\) stores the spanning tree of net \(n_i\). The \(while\) loop in its one iteration determines a branch to an uncovered pin, therefore, it usually has up to \(m - 1\) iterations (where \(m\) is the number of pins of \(n_i\)). The filling and retracing processes are similar to that of the Lee’s algorithm on a weighted grid. This work proposes a new function to determine the cost of any cell of the grid. For any cell \(v_j \in V\) whose preceding cell in the filling process is \(v_i\) and the edge between them is represented as \(e_{ij}\). The cost of \(v_j\) can be computed as follows:

\[
\text{cost}(v_j) = 1 + e^{u_{ij} - (c_{ij} - \beta)} + \text{cost}(v_i).
\]  

In Eq. (4), \(u_{ij}\) and \(c_{ij}\) represent the usage and capacity of the edge \(e_{ij}\) and the purpose of the exponential term is to avoid selection of paths through congested edges. When \((u_{ij} - (c_{ij} - \beta)) < 0\), the role of congestion is very limited and the algorithm find minimum length paths. However, when \((u_{ij} - (c_{ij} - \beta)) > 0\), then the role of

---

Fig. 3. Bounding region of MRF method for a net whose pins are \((p_0, p_1, p_2)\).
congestion costs becomes significant and the paths are determined w.r.t. minimum length as well as minimum congestion. The value of the parameter $C_1$ is initially set by the user but can be varied in the R&R process in order to build spanning trees that have minimum length and/or minimum congestion. At the start of the filling process, the cost values of all cells in $t_i$ are set to 0 and the cost values of the remaining cells in the bounded region is determined using the above mentioned equation. The filling process terminates when at least one of the following two conditions is reached: (a) the weights are assigned to all cells in the bounded region, or (b) a pin of the net which is not yet selected in $t_i$ is found. The retracing process, forms a branch to a pin of the net to any one node (or cell) of $t_i$.

3.3. R&R process

The R&R process component executes the R&R process to eliminate congestion from the solution of initial routing. Figure 5 shows the steps in an iteration of the R&R process. The function of each component is described in the following.

3.3.1. GT-based heuristic

In this work, GT is used to solve the problem of deciding which nets should be ripped-up and rerouted in order to eliminate congestion. The selection of nets in the R&R process is modeled as a game in which nets act as players. The set $N$ becomes the set of players of the game. Each net $n_i \in N$ has two pure strategies $S_i = \{\text{Yes, No}\}$. The strategy Yes means that $n_i$ should attempt to improve its spanning tree and strategy No means that $n_i$ should not attempt to improve its spanning tree. The players use mixed strategies to select their pure strategies. The mixed strategy of each player $n_i \in N$ is represented as $PR_i = (p_Y, 1 - p_Y)$, where $p_Y$ is the probability of selecting the strategy Yes and $1 - p_Y$ is the probability of selecting the strategy No. The Nash equilibrium (NE)\textsuperscript{12,13} of the game is reached when the tof of the solution becomes zero and at that point all nets want to stick to their No strategy ($p_Y$ becomes zero for all nets). The aim of the proposed GT-based heuristc is to eliminate congestion, however, it could be extended to multiple objectives.

\begin{verbatim}
Input: $P_i = \{p_0, p_1, ..., p_{m-1}\}$
Output: $t_i$: spanning tree of $n_i \in N$
1: Determine the bounded region
2: Randomly select a pin $p_x \in P_i$
3: while $t_i$ is not completed do
4: Filling process from $t_i$ to $p_x$, where $p_x \notin t_i \in P_i$
5: Retrace process to form a branch from $p_x$ to $t_i$ and add the branch to $t_i$
6: endwhile

Fig. 4. Procedure of MRF method.
\end{verbatim}
The main idea behind the proposed method of computation of mixed strategies is as follows: Each player wants to achieve two goals: (i) Its spanning tree becomes congestion free and (ii) its spanning tree should not be blocking the routing of any other net. In any iteration, the nets which are more likely to progress towards achieving their goals will have higher values of mixed strategies and hence, are more likely to go through the R&R operation.

Figure 6 shows the method which computes the mixed strategies of nets. The input contains two parameters $CT_1$ and $TP$. The parameter $CT_1$ acts as the weight of the term related to the overflow of the spanning trees and $TP$ is used to select the formula to compute the value of $p_Y$. The first formula of $p_Y$ assigns higher values to the nets whose pins enclose a smaller area as compared to the other nets and $p_Y$ of the nets which have no overflow is zero. The benefit of the first formula is that the nets that have congestion as well as require less time in rerouting should be preferred for the R&R operation as compared to others. The second formula of $p_Y$ assigns values based on the following principles: (i) the nets that have more overflow and fully used edges will have a higher $p_Y$ value, (ii) the weight of the overflow value in $p_Y$ can be changed using the parameter $CT_1$, and (iii) the nets that have not been ripped-up and rerouted since many iterations will have larger $p_Y$ values. The $p_Y$ values of the nets that have neither congested edges nor fully used edges is zero. The pseudo-code in Fig. 6 also shows that if none of the nets have any overflow then $p_Y$ values of all nets becomes zero. When the value of parameter $CT_1 \gg 1$, then the $p_Y$ values rely more on the overflow of the edges as compared to other factors. When $CT \approx 1$, then $p_Y$ values rely equally on overflow and number of fully used edges. When $CT1 \ll 1$, then $p_Y$ values rely more on the number of fully used edges.
The next step is to prepare a set $SS$ that contains the nets whose spanning trees should be rip-up and rerouted. In the selection of nets in $SS$, the $p_Y$ values of the nets act as their probabilities with which they could be selected in $SS$.

3.3.2. Apply R&R operation

The input of the component Apply R&R operation is the set $SS$. This work uses two types of R&R operations: (i) R&R Type A and (ii) R&R Type B. The Type A operation rips-up and reroutes one net at time, whereas, the Type B operation first rips-up two nets and then reroutes them. In both type of operations, the nets are ripped-up and rerouted completely. The R&R process is executed sequentially. The nets in $SS$ are divided into two subsets $SS_A$ and $SS_B$ such that $SS_B$ contains $RR1\%$ nets of $SS$ and $SS_A = SS - SS_B$. Furthermore, the number of nets in $SS_B$ should be even, otherwise, one element should be moved from $SS_B$ to $SS_A$. The nets in $SS_A$ go though the Type A operation and the nets in $SS_B$ go through the Type B operation. The proposed global router uses two types of R&R operations because it was

\[ \text{Input: } T: \text{ Spanning trees of all nets, } N: \text{ set of all nets, Parameters: } CT1 \in Z, TP \in \{True, False\} \]

\[ \text{Output: } PR = \{PR_0, PR_1, ..., PR_{N-1}\}: \text{ mixed strategies of all nets} \]

1. for each net $n_i \in N$ do
2. \quad $ofl(n_i) = \sum_{e_i \in t_i} overflow(e_i)$
3. \quad $fl(n_i) = \sum_{e_i \in t_i} full(e_i)$
4. \quad $area(n_i) = \text{area bounded by the pins (p_i) of } n_i$.
5. \quad $Iter(n_i) = \text{Number of iterations since } n_i \text{ was ripped-up and re-routed last time.}$
6. endfor
7. if $\sum_{n_i \in N} ofl(n_i) > 0$ then
8. \quad Normalize values in $ofl$, $fl$ and $Iter$.
9. \quad for each net $n_i \in N$ do
10. \quad \quad $p_Y(i) = \begin{cases} \frac{1}{area(n_i)} & \text{if } TP = 0, ofl(n_i) > 0, \\
11. \quad \quad \quad \quad CT1 \times ofl(n_i) + fl(n_i) \times (1 + Iter(n_i)) & \text{if } TP = 1, \\
12. \quad \quad \quad \quad 0 & \text{otherwise} \end{cases}$
13. \quad endfor
14. \quad Normalize $p_Y$ values
15. else
16. \quad for $i=0$ to $N - 1$ do
17. \quad \quad $p_Y(i) = 0$
18. \quad endfor
19. endif
20. for $i=0$ to $N - 1$ do
21. \quad $PR_i = \{p_Y(i), 1 - p_Y(i)\}$
22. endfor
23. return $(PR)$

Fig. 6. Method to determine the mixed strategies of all nets in $N$.

The next step is to prepare a set $SS$ that contains the nets whose spanning trees should be rip-up and rerouted. In the selection of nets in $SS$, the $p_Y$ values of the nets act as their probabilities with which they could be selected in $SS$.
observed by the authors in their experiments that convergence time can be reduced by using more than one type of R&R operation.

The R&R process is executed as follows: A net \( n_i \) is fetched from SS, if \( n_i \in SS_A \), then Type A operation is applied to it. If \( n_i \in SS_B \), then another net \( n_j \in SS_B \) is fetched from SS and Type B operation is applied to the nets \( n_i \) and \( n_j \). The main steps in both types of R&R operations are as follows: (i) copy the spanning tree(s) of the selected nets into temporary variables, (ii) completely delete (rip-up) the existing spanning tree(s), (ii) create new tree(s) using MRF method and (iv) compare the new tree(s) with the existing ones (i.e., the trees stored in temporary variables) and keep the better one(s). The comparison is required to ensure that the R&R process is moving towards congestion elimination with limited amount of hill-climbing. Figure 7 shows the function that compares two spanning trees \( t_i \) and \( t_i^* \) of a net \( n_i \), where \( t_i^* \) is the new spanning tree and \( t_i \) is the original spanning tree. The function returns true, if \( t_i^* \) is better than \( t_i \). In the comparison function, if the inferior solutions are always rejected then there is a high chance that our R&R process can get stuck into a local optima because of no hill-climbing. Therefore, the R&R operation occasionally allows acceptance of inferior solutions whose difference in the tof value with the original solution is not more than \(-NEG\), where \( NEG \) is a parameter which can be set by the user.

**Input:** \( G(V,E), t_i, t_i^* \): two spanning trees for a net \( n_i \), \( NEG \in Z^+ \)

**Output:** \( Y \in true, false \)

1: \( Y = false \)
2: \( A = \sum_{e_x \in t_i} overflow(e_x) \)
3: \( B = \sum_{e_x \in t_i} full(e_x) \)
4: \( C = |t_i| \)
5: \( D = \sum_{e_x \in t_i^*} overflow(e_x) \)
6: \( E = \sum_{e_x \in t_i^*} full(e_x) \)
7: \( F = |t_i^*| \)
8: \( r_{NEG} = \) a random integer between \(-NEG\) and 0
9: if \( D < A \) or \((D - A) \leq r_{NEG} \) then
10: \( Y = true \)
11: else
12: if \( D == A \) and \( E < B \) then
13: \( Y = true \)
14: else
15: if \( D == A \) and \( B == E \) and \( F < C \) then
16: \( Y = true \)
17: endif
18: endif
19: endif
20: return \( Y \)

Fig. 7. Method of comparing two trees \((t_i, t_i^*)\) of net \( n_i \).
3.3.3. Update parameter values

The proposed global router has two types of parameters: static parameters and self-adjustable (adaptable) parameters. The values of static parameters are set by the user and remains constant throughout the execution. The values of self-adjustable parameters change following an arithmetic progressions (APs) in the component Update parameter values. The AP of any parameter can be completely specified by three terms: (first term, last term and difference). The first term in its initial value, the last term is its maximum value and the difference is the amount by which it increments during self-adjustment. The user specifies the three terms of AP for each self-adjustable parameter. During the execution of global router, the component Update parameter values update the parameter value using their APs. In most of the self-adjusting parameters, when their value becomes equal to the last term of their AP, then their next value is the first term of their AP. Table 1 shows all parameters of the proposed global router and classifies them as static or self-adjustable. The parameter $T_m$ represents a threshold value for $tof$ value of the current iteration such that if its value becomes smaller than $T_m$, then the value of $BOX\_SIZE$ is set equal to its last term. The parameter $CT2$ represents the number of preceding iterations whose $tof$ values are used in adjusting the parameters values.

Figure 8 shows the procedure that is applied in each iteration of the R&R process in the component Update parameter values. The inputs are: Current iteration count and tof values of the current iteration and that of last $CT2$ iterations. In the first iteration (i.e., $i = 0$), the values of parameters are initialized and later on, the values of the parameters are adjusted based on overflow value of its preceding iterations. In the first iteration, the value of $TP1$ is assigned to zero, however, it changes to one based on the conditions mentioned in Fig. 8. The benefit of self-adjusting parameters is that they can help in exploring unique spanning trees for the nets. The proposed global router employs only MRF method of routing, and if a same method is applied to a net multiple times with same parameters values, then there is a high possibility that it returns a same solution every time. However, the chances of getting a different spanning tree for a net increases significantly by using different parameter values.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Reference</th>
<th>AP representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR1</td>
<td>Self-adjustable</td>
<td>Section 3.3.2</td>
<td>$RR1(\text{RR}_1,\text{RR}_f,\text{RR}_d)$</td>
</tr>
<tr>
<td>CT1</td>
<td>Self-adjustable</td>
<td>Section 3.3.1</td>
<td>$CT1(\text{CT}_1,\text{CT}_f,\text{CT}_d)$</td>
</tr>
<tr>
<td>TP1</td>
<td>One-time self-adjustable</td>
<td>Section 3.3.1</td>
<td>—</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Self-adjustable</td>
<td>Section 3.2</td>
<td>$\beta(\beta_i,\beta_f,\beta_d)$</td>
</tr>
<tr>
<td>BOX_SIZE</td>
<td>Self-adjustable</td>
<td>Section 3.2</td>
<td>BOX_SIZE(Box_SIZE, BOX_SIZE, BOX_SIZE, BOX_SIZE)</td>
</tr>
<tr>
<td>CT2</td>
<td>Static</td>
<td>Section 3.3.3</td>
<td>—</td>
</tr>
<tr>
<td>NEG</td>
<td>Static</td>
<td>Section 3.3.2</td>
<td>—</td>
</tr>
<tr>
<td>$T_m$</td>
<td>Static</td>
<td>Section 3.3.3</td>
<td>—</td>
</tr>
</tbody>
</table>
4. Experimental Results

The proposed global router was implemented using C++ and compiled using the Intel compiler for Linux. The code was executed on a virtual machine (VM) of the university’s cloud computing facilities. The VM has eight 2 GHz virtual processors, 16 GB of memory and RedHat Linux OS. It can execute up to eight concurrent threads. The proposed global router was implemented with the following parameter values: $RR_1 = (10, 60, 5), \ CT_1 = (1, 10, 1), \ BOX_SIZE = (1, \max\left(\frac{X_{\text{max}}}{2}, \frac{Y_{\text{max}}}{2}\right), 5)$ (where $X_{\text{max}}$ and $Y_{\text{max}}$ represent the horizontal and vertical grid sizes), $\beta = (1, 5, 1)$.

Input: $i$: current iteration, Overflow values of last $CT_2$ iterations: $\{tof(\bar{T}_i), tof(\bar{T}_{i-1}, \ldots, tof(\bar{T}_{i-CT_2})\}$

1: if $i = 0$ then
2: $RR_1 = RR_{1i}, CT_1 = CT_{1i}, BOX_SIZE = BOX_SIZE_i, \beta = \beta_i$
3: $TP_1 = 0$
4: else
5: $RR_1 = RR_1 + RR_{1d}$
6: if $RR_1 > RR_f$ then
7: $RR_1 = RR_{1i}$
8: endif
9: if $tof(\bar{T}_i) = tof(\bar{T}_{i-1})$ then
10: $CT_1 = CT_1 + CT_{1d}$
11: $TP_1 = 1$
12: endif
13: if $CT_1 > CT_f$ then
14: $CT_1 = CT_{1i}$
15: endif
16: if $tof(\bar{T}_i) < T_m$ then
17: $BOX_SIZE = BOX_SIZE_f$
18: else
19: if $tof(\bar{T}_i) = tof(\bar{T}_{i-1}) = \ldots = tof(\bar{T}_{i-CT_2})$ then
20: $BOX_SIZE = BOX_SIZE + BOX_SIZE_d$
21: if $BOX_SIZE > BOX_SIZE_f$ then
22: $BOX_SIZE = BOX_SIZE_f$
23: endif
24: $\beta = \beta + \beta_d$
25: if $\beta > \beta_f$ then
26: $\beta = \beta_i$
27: endif
28: endif
29: endif
30: endif

Fig. 8. Procedure to update the parameter values.
\(CT2 = 2\), \(NEG = 2\), \(T_m = 2\). The value of \(TP1\) do not need to be set as it is set internally by the component Update parameter values. The implementations of the proposed global router are represented as \(Proposed(x)\), \(x\) is the value of parameter \(NUM\text{.THREADS}\). In the experiments, \(x\) value was kept equal to 1, 4 and 8, because modern processors can usually execute four (dual-core) or eight (quad-core) concurrent threads.

The parameters were set based on the experimental observations to optimize both speed and solution quality. The value of \(RR1\) lies between 10 and 60, so that both types of R&R operations can be applied to the nets. The parameter \(CT1\) is used by the formula which computes the mixed strategies of nets. The mixed-strategy formula of each net contains two terms, the first one is the overflow of its spanning tree and the second one is the number of fully used edges in its spanning tree. The parameter \(CT1\) is the weight of the overflow term. When the value of \(CT1\) is large, then the nets which have large overflow also have large mixed strategies than the nets which have small overflow but large number of fully used edges. When the value of \(CT1\) is small, then the nets which have a small overflow but large number of fully-used edges also have large mixed strategies. If the value of \(CT1\) is too large, then nets having no-zero overflow will be only selected in the R&R operation. The value of \(CT1\) is varied so that nets having spanning trees of different characteristics can go through the R&R operation. The value of \(BOX\_SIZE\) was initially set to one, so as to speed-up the initial routing phase. The \(NEG\) value was set so as to allow small amount to hill-climbing. The value of \(T_m\) is set to a small value because when the R&R process is near completion, then a small number of nets participate in the R&R process which should be routed using maximum resources. It was also noticed that a different set of parameter values can produce solutions that are quite different in terms of wire-length and execution time. The results and binaries of our global router for the Linux 64-bit platform are available at the author’s website.\(^d\)

The test problems of the ISPD’98 suite\(^a\) were used to evaluate the performance of the proposed global router. Table 2 shows the characteristics of the problems in the ISPD’98 suite. Almost all heuristics use random number generators and therefore

<table>
<thead>
<tr>
<th>Test problem</th>
<th># of nets</th>
<th>Grid size</th>
<th>Test problem</th>
<th># of nets</th>
<th>Grid size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibm01</td>
<td>11507</td>
<td>64 × 64</td>
<td>ibm02</td>
<td>18429</td>
<td>80 × 64</td>
</tr>
<tr>
<td>ibm03</td>
<td>21621</td>
<td>80 × 64</td>
<td>ibm04</td>
<td>26163</td>
<td>96 × 64</td>
</tr>
<tr>
<td>ibm05</td>
<td>27777</td>
<td>128 × 64</td>
<td>ibm06</td>
<td>33354</td>
<td>128 × 64</td>
</tr>
<tr>
<td>ibm07</td>
<td>44394</td>
<td>192 × 64</td>
<td>ibm08</td>
<td>47944</td>
<td>192 × 64</td>
</tr>
<tr>
<td>ibm09</td>
<td>50393</td>
<td>256 × 64</td>
<td>ibm10</td>
<td>64227</td>
<td>256 × 64</td>
</tr>
</tbody>
</table>

\(^d\)https://sites.google.com/site/ufsresearch/ (posted on 30 December 2014).
their performance may become dependent upon the sequence of the random number generated by the generator. The sequence is controlled by the seed value of the generator. Therefore, the effect of different seed values should be considered in the experiments. The results of the proposed global router are not dependent on any particular seed value. In the experiments, each problem was solved for up to hundred times (i.e., have 100 trials) with a unique seed value. The proposed global router was compared with two recent routers: Box Router 2.0\textsuperscript{19} and Side-winder.\textsuperscript{20,21} The executable of Box Router 2.0 was obtained from its authors and executed on the same VM on which we executed our proposed global router. The parameters of Box Router 2.0 were set so as to optimize it for speed and obtain a valid solution in smallest possible time. The parameters values were set as follows: MULTI\_LAYER = 0, PREROUTING = 1, BOXROUTING = 1, BOXROUTING\_STEP = 1600, BOXROUTING\_ILP = 0, ILP\_SOLVER = 1, REROUTING = 1, REROUTING\_COUNT = 1000, REROUTING\_REPEAT = 0, REROUTING\_STEP = 100, REROUTING\_PUSH = 3, REROUTING\_STUCK = 5, MAZEROUTING\_MARGIN = 150, and RELAYERING = 0. Box Router 2.0 uses GNU Linear Programming Kit (GLPK)\textsuperscript{e} as an external ILP solver. It employs GLPK using its C++ application programming interfaces (APIs) which do not support MT. Therefore, Box Router 2.0 executed as a single-threaded program on the multi-core VM used in the experiments. The source code or executable of Side-winder is not publicly available, therefore, its published results\textsuperscript{20,21} were used in comparison.

In the global routing problem, the primary objective is to find a valid or congestion-free solution and the secondary objectives include minimization of wire-length and execution time. In our experiments, the proposed global router and Box Router 2.0 has solved all ten problems of the ISPD’98 suite. The Side-winder has

\begin{table}[h]
\centering
\caption{ToF of the solutions obtained from different global routers.}
\begin{tabular}{llll}
\hline
\textbf{Problem} & \textbf{Proposed(x), x \in \{1, 4, 8\}} & \textbf{Box Router 2\textsuperscript{19}} & \textbf{Side-winder\textsuperscript{20,21}} \\
\hline
ibm01 & 0 & 0 & 255 \\
ibm02 & 0 & 0 & 8 \\
ibm03 & 0 & 0 & 0 \\
ibm04 & 0 & 0 & 618 \\
ibm05 & 0 & 0 & 0 \\
ibm06 & 0 & 0 & 0 \\
ibm07 & 0 & 0 & 0 \\
ibm08 & 0 & 0 & 0 \\
ibm09 & 0 & 0 & 0 \\
ibm10 & 0 & 0 & 0 \\
\hline
\end{tabular}
\end{table}

\textsuperscript{e}https://www.gnu.org/software/glpk/ (accessed on 30 December, 2014).
solved up to seven problems. Therefore, the proposed global router is better than Side-winder in the most important goal of global routing which is routability.

Tables 4 and 5 report the results of the wire-length and runtime, respectively. The results of \textit{Proposed}(8) consists of three columns that are represented as median, Q1 and Q3. The medium column contains the medium of all the trials. The columns Q1 and Q3 contains the lower quartile or 25th percentile and upper quartile or 75th percentile of the results in different trials, respectively. Q1 and Q3 indicate the minimum and maximum values between which results of most of the trials exist. The results of \textit{Proposed}(4) and \textit{Proposed}(1) consist of only one column which contains the median of the trials. The last column has three or four sub-divisions and contains the gain which is expressed as the difference between the median result of the proposed router and the result of the Box Router 2.0 or Side-winder.

Table 4 shows that the solutions of the proposed global router (\textit{Proposed}(x), for \(x = 1, 4 \text{ or } 8\)) have wire-lengths better than that of Box Router 2.0 in up to seven test problems (i.e., ibm02, ibm03 and ibm06-to-ibm10). The Side-winder has obtained better wire-lengths in many test problems, however, it could not solve three test problems.

Table 5 shows that median as well as upper quartile of the trials of \textit{Proposed}(8) and \textit{Proposed}(4) are better than that of Box Router 2.0 in up to eight test problems (ibm02, ibm03 and ibm05-to-ibm10). In six test problems (ibm01-to-ibm03, ibm07, ibm09 and ibm10) the runtime of \textit{Proposed}(1) is better than or at most 2.6 s more than that of Box Router 2.0.

Tables 4 and 5 convey the following information about the comparison between the proposed global router (\textit{Proposed}(8) or (\textit{Proposed}(4)) and Box Router 2.0. In seven problems in which the proposed global router found solutions with smaller wire-length, it also found with better runtime. The parameters of Box router 2.0 were adjusted for speed, therefore, the results show that the proposed global router is faster than Box Router 2.0 in finding valid solutions. A different parameter values can be set in Box Router 2.0 which could improve its wire-length at the expense of increase in runtime. The Box Router 2.0 does not use MT because it employs an external ILP solver GLPK that does not support MT. The use of a faster ILP solver is a topic of future research for Box Router 2.0.18,19

Figure 9 shows the results of peak memory usage of the proposed global router. The results shows that the memory usage of the proposed global router depends directly on the wire-length and number of concurrent threads. Memory usage of the proposed global router with NUM_THREADS = 8 remains below 1.5 GB.

In addition to GT, the proposed global router also uses MRF and MT techniques to quickly converge to a valid solution. The proposed GT-heuristic has a key role in minimizing the runtime. Figure 10 shows that the absence of the proposed GT-based heuristic from the proposed global router causes an increase in its runtime in all problems.
Table 4. Wire-lengths of the proposed global router, Box Router 2.019 and Side-winder.20,21

<table>
<thead>
<tr>
<th>Problem</th>
<th>Proposed(8) Median ($x_1$)</th>
<th>Proposed(4) Median ($x_2$)</th>
<th>Proposed(1) Median ($x_3$)</th>
<th>Box Router 2.0 Median ($x_4$)</th>
<th>(x_5)</th>
<th>Gain ($x_4 - x_1$)</th>
<th>($x_4 - x_2$)</th>
<th>($x_4 - x_3$)</th>
<th>($x_5 - x_1$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibm01</td>
<td>66953.00</td>
<td>66717.00</td>
<td>67271.00</td>
<td>66875.00</td>
<td>66801.00</td>
<td>807.15</td>
<td>4082.50</td>
<td>2974.50</td>
<td>-1552.00</td>
</tr>
<tr>
<td>ibm02</td>
<td>1755164.50</td>
<td>174921.50</td>
<td>175683.00</td>
<td>175240.00</td>
<td>175388.00</td>
<td>829.00</td>
<td>4007.00</td>
<td>3859.00</td>
<td>-3859.00</td>
</tr>
<tr>
<td>ibm03</td>
<td>149076.00</td>
<td>149029.50</td>
<td>149182.00</td>
<td>149116.00</td>
<td>149054.50</td>
<td>655.00</td>
<td>1517.00</td>
<td>1517.00</td>
<td>-776.50</td>
</tr>
<tr>
<td>ibm04</td>
<td>174419.00</td>
<td>174133.50</td>
<td>174601.00</td>
<td>174224.00</td>
<td>174333.00</td>
<td>655.00</td>
<td>1517.00</td>
<td>1517.00</td>
<td>-776.50</td>
</tr>
<tr>
<td>ibm05</td>
<td>417756.50</td>
<td>417703.50</td>
<td>417792.00</td>
<td>417738.50</td>
<td>417730.00</td>
<td>843.50</td>
<td>825.50</td>
<td>825.50</td>
<td>-2508.50</td>
</tr>
<tr>
<td>ibm06</td>
<td>286025.00</td>
<td>285659.00</td>
<td>286448.00</td>
<td>285683.00</td>
<td>285849.00</td>
<td>3859.00</td>
<td>3859.00</td>
<td>3859.00</td>
<td>-3859.00</td>
</tr>
<tr>
<td>ibm07</td>
<td>373977.00</td>
<td>373737.50</td>
<td>374325.00</td>
<td>374057.00</td>
<td>374182.00</td>
<td>3819.50</td>
<td>3819.50</td>
<td>3819.50</td>
<td>-3819.50</td>
</tr>
<tr>
<td>ibm08</td>
<td>414076.50</td>
<td>413761.50</td>
<td>414457.00</td>
<td>413882.50</td>
<td>413930.00</td>
<td>7717.00</td>
<td>7717.00</td>
<td>7717.00</td>
<td>-7717.00</td>
</tr>
<tr>
<td>ibm09</td>
<td>421720.50</td>
<td>421544.00</td>
<td>422112.00</td>
<td>421721.50</td>
<td>421784.00</td>
<td>9876.50</td>
<td>9876.50</td>
<td>9876.50</td>
<td>-9876.50</td>
</tr>
<tr>
<td>ibm10</td>
<td>593544.50</td>
<td>593095.00</td>
<td>594020.00</td>
<td>593404.00</td>
<td>593825.00</td>
<td>9876.50</td>
<td>9876.50</td>
<td>9876.50</td>
<td>-9876.50</td>
</tr>
<tr>
<td>Total gain</td>
<td>34814.50</td>
<td>35567.50</td>
<td>35395.00</td>
<td>35567.50</td>
<td>35395.00</td>
<td>16265.00</td>
<td>16265.00</td>
<td>16265.00</td>
<td>-16265.00</td>
</tr>
</tbody>
</table>

Table 5. Runtime of the proposed global router and Box Router 2.019

<table>
<thead>
<tr>
<th>Problem</th>
<th>Proposed(8) (s)</th>
<th>Proposed(4) (s)</th>
<th>Proposed(1) (s)</th>
<th>Box Router 2.0 (s)</th>
<th>Gain (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibm01</td>
<td>5.55</td>
<td>4.93</td>
<td>6.39</td>
<td>4.60</td>
<td>5.25</td>
</tr>
<tr>
<td>ibm02</td>
<td>4.60</td>
<td>4.92</td>
<td>5.06</td>
<td>4.66</td>
<td>8.29</td>
</tr>
<tr>
<td>ibm03</td>
<td>2.32</td>
<td>2.05</td>
<td>2.60</td>
<td>2.57</td>
<td>5.04</td>
</tr>
<tr>
<td>ibm04</td>
<td>62.08</td>
<td>51.50</td>
<td>72.81</td>
<td>58.72</td>
<td>57.43</td>
</tr>
<tr>
<td>ibm05</td>
<td>2.46</td>
<td>2.39</td>
<td>2.71</td>
<td>4.53</td>
<td>17.10</td>
</tr>
<tr>
<td>ibm06</td>
<td>8.87</td>
<td>8.25</td>
<td>9.74</td>
<td>9.85</td>
<td>18.65</td>
</tr>
<tr>
<td>ibm07</td>
<td>7.70</td>
<td>6.98</td>
<td>8.70</td>
<td>8.29</td>
<td>15.83</td>
</tr>
<tr>
<td>ibm09</td>
<td>8.80</td>
<td>8.15</td>
<td>9.77</td>
<td>10.17</td>
<td>20.39</td>
</tr>
<tr>
<td>ibm10</td>
<td>15.45</td>
<td>13.26</td>
<td>17.11</td>
<td>16.81</td>
<td>33.12</td>
</tr>
<tr>
<td>Total gain</td>
<td>47.83 s</td>
<td>42.69 s</td>
<td>35395.00</td>
<td>47.83 s</td>
<td>35395.00</td>
</tr>
</tbody>
</table>
5. Conclusion

In this paper we presented a GT-based global router that employs MRF method and GT-based R&R process. In the GT-based R&R process, the nets act as players and each net has two pure strategies: (i) Attempt to improve its spanning tree and (ii) do not attempt to improve its spanning tree. The nets use mixed strategies to select any particular pure strategy. A method is proposed for the determination of mixed strategies. The nets that have selected to improve their spanning trees will go through the R&R operation. The proposed global router was implemented using C++ and the benchmarks of the ISPD’98 suite were used in performance evaluation. The proposed global router was compared with two recent ILP-based global routers,

![Fig. 9. Memory usage of the proposed global router.](image1)

![Fig. 10. Increase is runtime after removing GT-based selection of nets from the proposed global router.](image2)
which are: (i) Box Router 2.0 and (ii) Side-winder. The proposed global router with MT is faster than Box Router 2.0. The proposed global router has solved more problems as compared to Side-winder, therefore, it is better than Side-winder in terms of routability. The experimental results show that the proposed GT-based heuristic is effective in reducing the runtime of a simple maze router.

Acknowledgments

The authors wish to acknowledge King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia for all the support.

References


