INTEGRATION, the VLSI journal 53 (2016) 115–125

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

A stochastic evolution algorithm based 2D VLSI global router

Sadiq M. Sait [a], Umair F. Siddiqi [b,*]

[a] Department of Computer Engineering, and Center for Communications and IT Research, Research Institute, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia
[b] Center for Communications and IT Research, Research Institute, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia

ARTICLE INFO

Article history:
Received 7 July 2015
Received in revised form
22 December 2015
Accepted 22 December 2015
Available online 30 December 2015

Keywords:
Global routing
Discrete optimization
Stochastic evolution (StocE) algorithm
Maze routing

ABSTRACT

In this work we present a global router that uses the features of the Stochastic Evolution (StocE) algorithm to perform the rip-up and re-route (R&R) process. The unique features of the proposed global router are its simple design and ability to produce good results. It has two main components: (i) maze routing with framing (MRF) that act as a means of routing the nets; and (ii) a StocE algorithm that controls the R&R process and adjusts the parameters of the MRF method. This work shows that the StocE algorithm can perform the net selection in the R&R process and also adjust the parameter values of the MRF method in order to successfully solve 2D global routing problems. The performance of the proposed global router on the ISPD98 and ISPD2008 benchmarks was found to be better than some of the existing global routers.

1. Introduction

Global routing is an important step in the physical design phase of integrated circuits and is an NP-hard problem [1,2]. Its aim is to determine approximate routing paths for the nets in order to make connections between components. The global routers consider the routing region as a two-dimensional (2D) rectangular array of uniform grid cells. The components are placed on the grid and occupy one or more cells. The boundaries of the grid cells indicate their edges with the neighboring grid cells. Each edge has a fixed capacity which is equal to the maximum numbers of nets that can pass through it. An edge is considered congested if the number of nets passing through it exceeds its capacity. The primary goal of global routing is to route all nets such that the capacity constraints of the edges are not violated. A solution of the global routing problem that has no congested edges is called a valid solution. The secondary objective of global routing is to minimize the wire-length as smaller wire-length solutions are cost-efficient.

Recently, 3D global routing has also emerged as a popular and important physical design problem. It has two or more routing layers. Each layer allows the nets to be laid down in either horizontal or vertical direction. The nets can connect between layers using vias. The 3D global routing is usually performed by first transforming it into a 2D global routing problem. In the transformed 2D problem, the edges have capacities which is equal to the sum of their capacities in all layers. The 2D transformed problem can be solved using 2D global routing methods. The solution of a 2D global routing problem can then be transformed back into a 3D global routing solution using layer assignment. It has been proven that a valid (or congestion-free) solution of 2D global routing problem can be successfully transformed into a valid 3D global routing solution [3]. Therefore, the role of 2D global routing will remain important even in the future. Routability driven (RD) placement is another recent advancement in physical design. In RD placement the global routing process is combined with the placement process. The RD placement process needs global routers which are simple and fast in addition to being capable of producing good results [4–6].

The global routing process usually consists of three steps which are: (i) ordering of nets; (ii) generation of initial solution; and (iii) rip-up and re-route (R&R) process. The R&R process eliminates congestion and/or minimizes wire-length of the initial solution. Modern global routers are different from each other in their implementation of these steps. The R&R process is similar to an optimization process whose objective is to minimize congestion of the solution. Non-deterministic (ND) and evolutionary algorithms (EAs) are popular for solving combinatorial optimization problems. However, very small number of global routers use ND and EAs [7] to minimize congestion.

In the last 10 years, many global routers have been proposed but none of them have presented any significant improvement in the employment of ND and EAs to solve the global routing

* Corresponding author. Tel.: +966 13 860 8292; fax: +966 13 860 2215.
E-mail addresses: sadiq@kfupm.edu.sa (S.M. Sait), ufarooq@kfupm.edu.sa (U.F. Siddiqi).

http://dx.doi.org/10.1016/j.vlsi.2015.12.007
0167-9260/

problem. The TIMBERWOLF [8] and SILK [9] were proposed more than two decades ago and have adopted Simulated Annealing (SA) and Simulated Evolution (SimE) algorithms to model their R&R process. However, they can solve only small size of problems of certain layout types. In recent past, NCTU-GR [10] has improved the goodness function of the SimE algorithm which was used by SILK. The R&R process in NCTU-GR uses a simple design of SimE algorithm with two-stage cost function based maze routing. In NCTU-GR, the design of SimE algorithm remains insufficient to solve the R&R problem on its own and without the aid of a sophisticated routing method. The aim of our work is to propose an EA (such as Stochastic Evolution) that has enhanced design so that it can solve the R&R problem using a simple widely known routing method (such as maze routing). This work proposes new designs of the different functions of the StocE algorithm in-order to solve the R&R problem.

This paper proposes a Stochastic Evolution (StocE) algorithm [7,11] based global router. It uses the StocE algorithm to control the tuning of parameters and selection of nets in the R&R process. It uses only one method of routing which is maze routing with framing (MRF) [12]. It has lesser components as compared to the contemporary global routers. The performance of the proposed global router was evaluated on the ISPD98 [13,14] and ISPD2008 [15] benchmarks and compared with some recent global routers. The results show that its solution quality and runtime are competitive to the recent global routers.

This paper is organized as follows. The next section shows the relevant previous work. Section 3 describes the global routing problem; the StocE algorithm and its comparison with other ND and EAs. Section 4 describes the proposed global router. In Section 5, the experimental results are discussed and their analysis is provided. The last section contains the conclusion and future work.

2. Related previous work

This section presents some existing global routers which use evolutionary/stochastic algorithms. Sechen et al. proposed TIMBERWOLF [8] layout tool for the placement and routing on the standard cells. It uses simulated annealing (SA) in both placement and global routing. The global routing in TIMBERWOLF consists of two steps. The first step is the initial routing of the nets in which they are assigned to horizontal routing channels. At the end of first step, all nets whose assignment can be changed to adjacent channels are identified as switchable nets. In the second step, the channel densities are attempted to be reduced by changing the assignment of switchable nets. The TIMBERWOLF has the restriction that it only works on the standard cell layout.

Lin et al. proposed SILK [9] which consists of two steps. The first step performs the initial routing of the nets and the second step execute the R&R to eliminate congestion. They used simulated evolution (SimE) to select nets in the R&R operation. The routing of nets is performed using a modified Lee's algorithm. They applied their algorithm to the standard cell and switch-box layouts. SILK was applied to small size problems and exhibited very long runtime as compared to the recent routers. The authors state that the application of their global router to the full custom design requires further enhancement because the global routing on a full-custom layout is more complex than other types of layouts.

Dai et al. proposed NCTU-GR [10] which uses SimE to select nets in the R&R process. They used FLUTE [16] to generate an initial routing solution. It has two R&R processes. The first R&R process is simpler and faster than the second one. The first R&R process consists of circular fixed ordering monotonic routing, in which the congested nets are sorted in the decreasing order of their wire-lengths and the re-routing was performed using monotonic routing. The second R&R process uses the SimE algorithm [7] to select nets for the R&R operation. The goodness of nets was determined based on three factors: (1) number of congested edges, (2) wire-length, and (3) number of vias. They proposed a two-stage cost function for maze routing which is an enhanced form of the history based cost function used in many global routers such as Archer [17]. The two-stage cost function based maze routing was very efficient and the authors showed that it can solve complex problems even with the traditional R&R method. The role of SimE-algorithm was to reduce the runtime of the R&R process. NCTU-GR has been upgraded to a newer version as NCTU-GR 2.0 [18] in which the SimE algorithm was completely eliminated.

Perez et al. proposed to use genetic algorithm (GA) with Lee's algorithm in order to minimize its search space [19]. The GA determined a minimum subset of nodes on the routing grid that contains one or more paths between the source and destination nodes. The nodes were selected based on their likelihood to be part of the shortest path between the two nodes. Lee's algorithm searched for the path using only the nodes selected by the GA. Their work reduced the search space of Lee's algorithm by 30–50% but exhibited long convergence time as compared to the recent global routers. Their contribution also reduced the memory requirement of Lee's algorithm.

3. Definitions

3.1. Global routing problem

The global routing problem is modeled using a 2D grid-graph G of dimensions $X_{max} \times Y_{max}$ [20]. A set V is defined that contains the vertices and a set E contains the edges of the grid-graph. Each vertex $v_i \in V$ corresponds to a particular rectangular region (or cell) of the chip, and each edge $e_{ij} \in E$ corresponds to a boundary between adjacent vertices v_i and v_j. The capacity of any edge $e_{ij} \in E$ is represented by c_{ij}, which indicates the maximum number of nets or wires that can pass through it. The actual number of nets that are passing through e_{ij} is called its demand and is represented as u_{ij}. The set $N = \{n_0, n_1, \ldots, n_{m-1}\}$ contains the nets that should be routed on the grid-graph. Each net $n_i \in N$ has a set P_i which contains its pins. A pin corresponding to a vertex on the grid-graph. A net can be routed on the grid-graph by means of a spanning tree which connects all of its pins. The spanning tree of the net n_i is represented by t_i and $t_i \subseteq E$. The set T stores the spanning trees of all nets. It is also called as the solution of the global routing problem.

The primary objective of global routing is to route all the nets while making sure that the capacity constraints of the edges are not violated, i.e., $u_{ij} \leq c_{ij}, \forall e_{ij} \in E$. For any edge e_{ij}, if its usage (u_{ij}) becomes greater than its capacity (c_{ij}) then its is called as congested. The congested edges contribute to the total overflow. For any edge e_{ij}, its overflow can be expressed as $overflow(e_{ij})$ and its value can determined using the following equation:

$$overflow(e_{ij}) = \begin{cases} u_{ij} - c_{ij} & \text{if } u_{ij} > c_{ij} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

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

$$tof(T) = \sum_{e_{ij} \in E} overflow\ (e_{ij}) \quad (2)$$

The total wire-length determines the resource efficiency or manufacturing cost of a valid solution. It can be computed as

follows:

$$tot(T) = \sum_{t_i \in T} |t_i| \quad (3)$$

3.2. Combinatorial optimization problem of the R&R Process

In this work, the R&R process of global routing is considered as a combinatorial optimization problem (COP) [21] which can be represented as a tuple ($\mathbf{U}$,f), where $\mathbf{U}$ is a finite set of possible solutions of the global routing problem, i.e., $\mathbf{U} = \{T_0, T_1, \ldots T_{K-1}\}$, and $T_i \neq T_j$, for $i \neq j$, and $T_i = \{t_0, t_1, \ldots, t_{N-1}\}$, where $T_i \in \mathbf{U}$ and t_j is the spanning tree of net $n_j \in N$ ($j = 0$ to $N-1$). The function f is the cost function that maps each element in set $\mathbf{U}$ onto the set of integers and defined as follows:

$$f(T_i) = overflow(T_i), \quad \text{where } T_i \in \mathbf{U} \quad (4)$$

The goal of COP is to find a solution T_{opt}, such that $f(T_{opt}) < f(T_i)$, $\forall T_i \in \mathbf{U}$. In the global routing problem, a solution becomes valid when its overflow becomes zero.

3.3. StocE algorithm

The StocE algorithm is a general purpose non-deterministic algorithm [7,11]. It resembles a biological evolutionary process in which the solution eliminates its bad characteristics as generation evolves. Fig. 1 shows the StocE algorithm. The input are T, which is a complete but invalid solution of global routing problem; and two parameters p_0 and R, that control the number of iterations. The optimization loop has two main functions: Perturb and Update. The Perturb function is responsible for replacing bad spanning trees of the nets with better ones and Update function changes the values of parameters. The Perturb function consists of two steps. The first step determines the movable elements and the second step executes one or more number of moves using the movable elements in order generate a new solution (Fig. 2). A movable element refers to a net whose spanning tree should be changed in order to improve the solution. The move can be of two types: simple and compound. A simple move is defined as ripping-up and re-routing a net. A compound move considers two nets at a time and it comprises of the following two steps: (i) ripping-up the nets and, (ii) re-routing them one after another. A detailed discussion on StocE algorithm is present in [7,11]. The proposed StocE algorithm will be described in detail in the latter sections.

Input: $T,\ p_0,\ R$
Output: T_{opt}
1: $T_{opt} = T, Cost_{opt} = f(T), Cost_{curr} = f(T)$
2: $p = p_0, \rho = 0$
3: **while** $\rho > R$ **do**
4: $\quad Cost_{prev} = Cost_{curr}$
5: $\quad M = FIND\ MOVABLE(T, p)$
6: $\quad T = PERTURB(T, p, M)$
7: $\quad Cost_{curr} = f(T)$
8: $\quad UPDATE(p, Cost_{prev}, Cost_{curr})$
9: $\quad$ **if** $Cost_{curr} < Cost_{opt}$ **then**
10: $\quad\quad Cost_{opt} = Cost_{curr}, T_{opt} = T$
11: $\quad\quad \rho = \rho - R$
12: $\quad$ **else**
13: $\quad\quad \rho = \rho + 1$
14: $\quad$ **end if**
15: **end while**
16: **return** T_{opt}

Fig. 1. StocE algorithm.

Input: $T,\ p,\ M$
Output: T_{opt}
1: **for** each element $m \in M$ **do**
2: $\quad T' = MOVE(T, m)$
3: $\quad Gain(m) = tof(T') - tof(T)$
4: $\quad$ **if** $Gain(m) > RANDINT(-p, 0)$ **then**
5: $\quad\quad T = T'$
6: $\quad$ **end if**
7: **end for**
8: **return** T

Fig. 2. Perturb operation of StocE algorithm.

3.4. Comparison of the StocE algorithm with others

The ND and EAs can be divided into two types: (i) population-based and (ii) single-solution-based. The population-based algorithms work on a set of solutions while the others work on only one solution. The GAs are the most popular population-based EAs [7]. The population-based algorithms consume more memory than single-solution-based algorithms. For example, to store an edge of a spanning tree, which can be represented by four coordinates (x_1, y_1, x_2, y_2) we require 16 bytes. For the problem adaptec5 (a benchmark used in our comparison), GA with a population size of 200 will require an estimated space of over 70.0 GB. Similar, the requirements for newblue7 is around 120 GB per population. Clearly, population based global routers will need an unreasonable amount of memory to perform computations. The performance of GAs is not good when population size is small. Therefore, single-solution-based algorithms are more suitable to handle such problems.

The problems whose solutions are memory intensive should be solved using single-solution-based algorithms. Some examples of single-solution-based algorithms include: Simulated Annealing (SA), Simulated Evolution (SimE) and StocE [7]. The features of the StocE algorithm that distinguishes it from the other single-solution-based algorithms are described below.

The SA algorithm can take only simple moves, whereas the SimE and StocE algorithms can execute simple as well as compound moves. In global routing, a simple move refers to ripping-up and re-routing one net and a compound move refers to ripping-up two or more nets and then re-routing them. In SA algorithm, the probability of hill-climbing decreases with runtime, whereas, in StocE algorithm, the probability of hill-climbing does not decrease and the solution can escape local minima with the same probability as in the initial iterations. The application of SA on different problems showed that its runtime is usually more than other algorithms.

In SimE and StocE algorithms a move refers to generating a new spanning tree for a net. In SimE algorithm, the move operation consists of generating several alternate spanning trees through trial-mutations and selecting the spanning tree which is best according to some defined criterion. On the other hand, the move operation in StocE algorithm generates only one spanning tree which could be accepted or rejected based on a probability function. The generation of several alternate spanning trees is becoming redundant due to the development of efficient routing methods that can produce good quality spanning trees for any given demand and capacity of edges. The global routing problem contains millions of nets and each net could have hundreds of pins, therefore, the computation of spanning tree can be considered as a time consuming operation. The StocE algorithm's move operation with an efficient routing method is expected to be faster than the move operation of the SimE algorithm.

3.5. Multi-pin net decomposition

The degree of a net is the number of its unique pins. The nets which have degree two can be connected using shortest path algorithms such as Dijkstra's algorithm, A* search. However, the nets of degree three or more are difficult to route. Therefore, modern global routers break the nets that have degree three or more into several pairs of pins. These pairs are called as subnets. The decomposition of a multi-pin net into subnets can be performed with the help Rectilinear minimal Steiner tree (RMST) packages such as Flute [16]. The net-decomposition can be performed w.r.t. minimum wire-length or congestion. The net-decomposition w.r.t. minimum wire-length is the most common type of net decomposition.

3.6. Properties of nets

In this work, each net is associated with three properties. These properties are used in the computations that select the nets for the R&R operation. The three properties of any net n_k are as follows: (i) $\#MST_k$, which indicates the length of the minimum spanning tree (MST) of net n_k; (ii) $\#t_k$, which indicates the size of the current spanning tree of n_k; and (iii) I_k, which indicates the most recent iteration in which n_k went through the R&R operation. The value of $\#MST_k$ can be obtained using a software tool such as Flute [16].

4. Proposed global router

The proposed global router has two main steps which are as follows: (ii) initial routing of nets and (ii) StocE-based R&R process. The first step routes all the nets and the second step removes any congestion which may exists at the end of the first step. The proposed global router is made up of the following components: (i) Ordering of nets; (ii) MRF method; and (iii) StocE algorithm. The components Ordering of nets and StocE algorithm are used exclusively by the first and second steps, respectively. The component MRF method is shared by both steps. This section first describes the working of the two steps and then the functioning of each component in detail.

Fig. 3 shows the sub-steps that constitutes the first step of routing in the proposed global router. The input is the set of nets (N) and the output is a solution that has spanning trees of all nets. In Fig. 4, the first step is Ordering of nets which is performed by the component 'Ordering of nets'. The next step consists of a loop

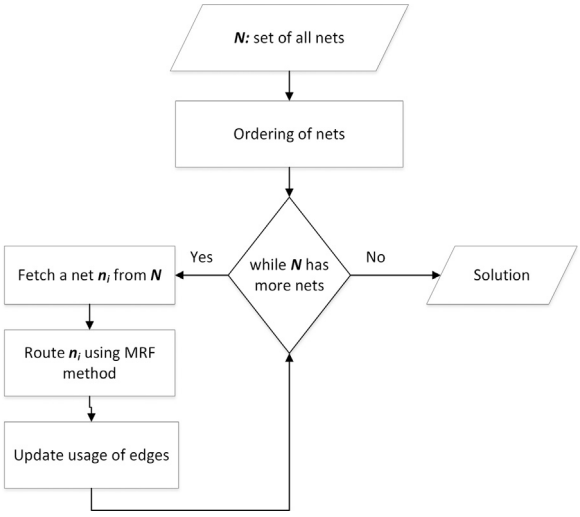

Fig. 3. Flow of execution in the first step 'Initial routing of nets'.

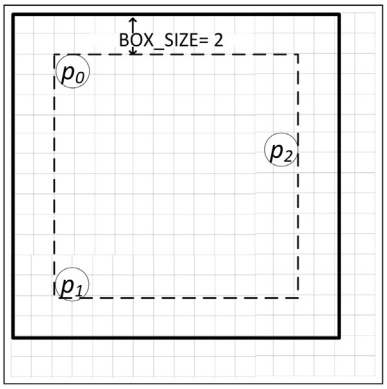

Fig. 4. Illustration of the frame of MRF when $BOX\_SIZE = 2$ and the net has three pins $\{p_0, p_1, p_2\}$.

which routes the sorted nets. The nets are routed using the MRF method. In initial routing, the MRF method could employ net decomposition using the component 'Decomposition of multi-pin net'. After each net is routed, the usage of the edges used by that net should be updated so that the next nodes can be routed using up-to-date usage of the edges. The second step consists of the StocE algorithm whose Perturb operation performs the R&R of some nets and Update operation modifies the parameter values of the StocE algorithm and that of MRF routing method. The MRF method in the R&R process do not use net-decomposition because experiments revealed no benefit of net-decomposition w.r.t. wire-length in the R&R process. The remaining portion of this section describes the components of the proposed global router.

4.1. Ordering of nets

Ordering of nets is considered necessary in sequential maze routing and useful in parallel routing. There are many methods of ordering the nets and this work uses the method to order the nets in the ascending order of the area bounded by their pins. This ordering is based on the observation that the nets which enclose a smaller area have lesser number of alternate spanning trees as compared to the nets that enclose a larger area. In this step, the nets in N are sorted and many nets change their positions and set N becomes an ordered set of nets. After the ordering of nets, in set $N = \{n_0, n_1, \ldots, n_{m-1}\}$, for any $j > i$, n_j has a larger bounding box as compared to n_i.

4.1.1. MRF routing

This work executes MRF method by employing Lee's algorithm on a weighted grid [12]. The routing is restricted to a smaller region whose size is determined by the parameter *BOX_SIZE* [12]. Fig. 4 shows an example frame of MRF. In case of subnets, the frame size is calculated using the two pins of the subnet and parameter *BOX_SIZE*. The MRF method used in this work is illustrated in the following with the help of an example.

Let us consider a net n_k which has up-to m pins (where all pins lie at different positions in the 2D grid graph) that are represented as $P_k = \{p_0, p_1, \ldots, p_{m-1}\}$. The spanning tree of n_k will be stored in t_k. The first step of the MRF method is to determine a routing region as already described. The MRF method executes Lee's algorithm on a weighted grid method [12] to determine the spanning tree of n_k.

Lee's algorithm on a weighted grid consists of filling and retrace steps. The filling step determines the weights of all cells in the routing region. The filling process assigns weights to the cells as follows: (i) the cells which exists in t_k will have zero weight; and (ii) the weight of any other cell can be determined using the

following formulas:

$$weight(v_j) = 1 + \frac{penalty}{1 + e^{-\beta(u_{ij} - c_{ij})}} + weight(v_i) \quad (5)$$

$$weight(v_j) = 1 + e^{u_{ij} - (c_{ij} - \beta)} + weight(v_i) \quad (6)$$

Either of the above equations can be used to determine the weights of the cells. The above equations determine the weight of a cell v_j whose preceding cell is v_i in the filling process. FastRoute [3,22] proposed to use the formula in Eq. (5) to quickly remove overflows in huge size problems. NCTU-GR [10] also uses the formula of Eq. (5) to fill the grid cells. However, in small and medium size problems, the use of Eq. (5) either yields longer wire-length or consumes too much runtime. FastRoute 2.0 [22] did not produce good results on small to medium size problems and NCTU-GR [10] was applied only to huge size problems. Therefore, for small and medium size problems, this work proposes a modification in the formula which is shown in Eq. (6). The parameter *penalty* in Eq. (5) represents the penalty that should be associated with the congestion of edges. If its value is larger, then the weights of congested edges becomes very large. In Eqs. (5) and (6), u_{ij} and c_{ij} represent the usage and capacity of the edge e_{ij}. The parameter β in Eq. (5) refers to the slope and a steeper slope means a more sudden change in the weights of the congested and non-congested edges. In this work, a small slope value is used in initial routing and in the initial R&R iteration, but the latter iteration in the R&R process use a steeper slope. In Eq. (6), a large value of parameter β assigns more weights to the edges that are congested or near congestion. When β has a value such that $(u_{ij} - (c_{ij} - \beta)) < 0$, then the weights are assigned in order to find a spanning tree of minimum wire-length. Whereas, when β has a value such that $(u_{ij} - (c_{ij} - \beta)) > 0$ then the spanning tree contains less congested edges. The value of β 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. The value of *penalty* is kept fixed in this work. The steps of the MRF routing method are outlined in the following.

1. Randomly select a pin p_x from P_k.
2. Insert p_x into t_k.
3. Perform the filling process in which all nodes in t_k act as source nodes. The termination condition of the filling process is the discovery of a node p_y that satisfies the following two conditions: $p_y \in P_k$ and $p_y \notin t_k$.
4. Perform the retrace process that starts from p_y and terminates at any node $p_z \in t_k$.
5. Add the path (or branch) obtained from the above retrace process into t_k.
6. Repeat the steps 3–5 unless t_k covers all pins in P_k.

The above-mentioned method can determine a spanning tree for a net which has two or more number of pins (i.e., whose degree is two or more). However, most of the existing global routers use multi-pin net decomposition. In multi-pin decomposition, the nets that have degree three or more are decomposed into subnets of degree two. Lee's algorithm can be applied to nets as well as subnets. The only difference is that in case of subnets the steps 3–5 do not require any repetition. This work uses multi-pin decomposition for minimum wire-length in the initial routing phase of small to medium size problems. The experiments on small to medium size problems showed that smaller wire-length can be achieved using MRF method with net-decomposition. However, in case of problems that have huge number of nets, the use of net decomposition w.r.t. minimum wire-length can significantly increase the congestion whose minimization becomes difficult in the R&R process.

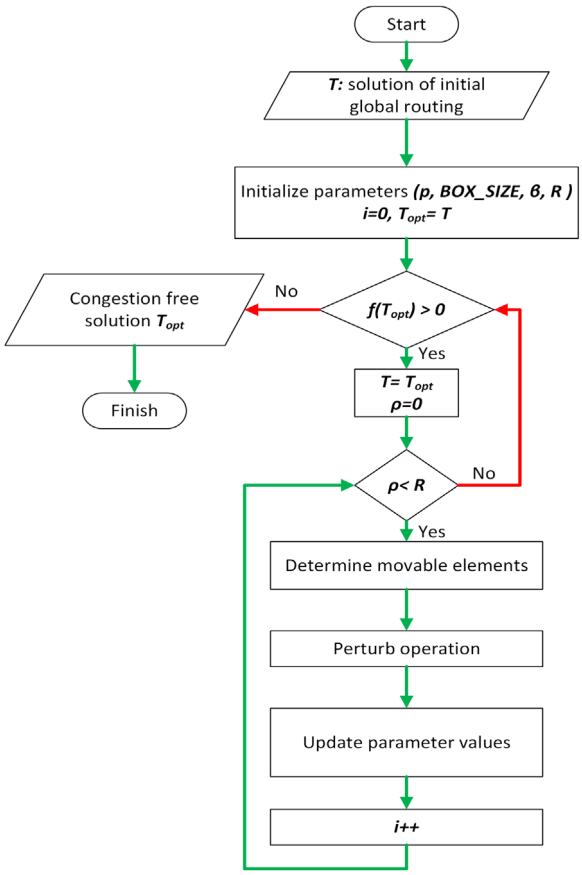

Fig. 5. Proposed StocE algorithm based R&R process.

4.2. StocE algorithm based R&R process

In this work, the R&R process is modeled as a COP which is solved using StocE algorithm. Fig. 5 shows the proposed StocE algorithm which is tailored to solve the R&R problem of global routing. The algorithm has four parameters (*BOX_SIZE*, β, *p*, *R*, and *S_OFL*). The parameters *BOX_SIZE* and β are already described in Section 4.1.1 in the description of MRF method. The remaining two parameters *R* and *p* are the same as in the standard StocE algorithm [12]. The parameter *R* is used to decide the number of iterations of the optimization loop and *p* controls the amount of hill-climbing (i.e, serves the same purpose as in the standard StocE algorithm). The proposed StocE-algorithm stores two solutions which are current solution (T) and optimal solution found so-far (T_{opt}). It also has two nested loops. The optimization is performed by the inner loop, however, the role of the outer loop is to reset the current solution to the best solution and to initiate another cycle of the inner optimization loop if the solution still has overflow. The variable *i* in Fig. 5 stores the iteration count of the R&R process. The variable ρ is used to control the number of iterations of the inner (or optimization) loop. The optimization loop continues until ρ remains smaller than *R*. The StocE algorithm has three main operations: (a) Determine movable elements, (b) Perturb, and (b) Update. The nets that are causing congestion are selected or marked as movable elements. The *Perturb* operation generates new spanning trees for the movable elements in-order to reduce congestion. The *Update* operation changes the parameter values of the StocE algorithm and MRF method. The following text describes the main operations of the StocE algorithm in detail.

4.2.1. Determine movable elements

The nets that are causing congestion are considered as movable elements and takes part in the *Perturb* function of StocE. In real problems, there could be many nets that are causing congestion. Therefore, a subset of nets should be selected to go through the Perturb function of StocE. The selected nets will be stored in a selection set *SS*. The method proposed in this work to select the nets for the Perturb function consists of the following steps:

1. Compute the probabilities of all nets.
2. The nets which have non-zero and high probability values are more likely to be selected in *SS*.
3. The nets in *SS* will act as movable elements in the Perturb function.

The probabilities of the nets can be computed as follows. Each net $n_k \in N$ whose spanning tree is t_k (where $t_k \neq null$) compute values of three variables $(of_k, Len_k, Iter_k)$ as follows:

$$ofl_k = \sum_{e_j \in t_k} overflow(e_j) \tag{7}$$

$$Len_k = \frac{\#t_k}{\#MST_k} \tag{8}$$

$$Iter_k = i - I_k \tag{9}$$

$$Pb_k = ofl_k \times Len_k \times Iter_k \tag{10}$$

Eq. (10) computes the probability Pb_k of the net n_k with which it could be selected in *SS*. After the probability (Pb_k) values of all nets are determined then they are normalized so that the probability values lie within the range of 0–1. The final step uses the probability values to select the nets for the R&R operation in the set *SS*.

4.2.2. Perturb operation

The perturb operation uses the nets selected in *SS* to form a new solution. Fig. 6 shows the proposed perturb operation. The input are the selection set *SS*, the current solution *T* which contains spanning trees of all nets, parameter *p* that is used to control the amount of hill climbing, and parameter *BOX_SIZE* that is used to determine the size of the bounding box in the MRF method.

Input: SS: movable elements ; $T = \{t_0, t_1, ..., t_{m-1}\}$: current solution, $p \in Z^+$, BOX_SIZE
Output: T: solution after the Perturb operation
1: permute the order of nets in SS
2: $T' = T$
3: **while** SS has elements **do**
4: Fetch a net n_i from SS
5: Find and fetch another net n_j from SS, such that the bounding boxes of n_i and n_j intersect each other.
6: **if** n_j is found **then**
7: Rip-up the nets n_i and n_j in any order (i.e. $T' = T' - \{t_i, t_j\}$)
8: Re-route the net n_i and n_j using the MRF method in any order (i.e. $T' = T' \cup \{t_i, t_j\}$)
9: **else**
10: Rip-up the net n_i (i.e., $T' = T' - t_i$)
11: Re-route n_i using the MRF method (i.e. $T' = T' \cup t_i$)
12: **end if**
13: **if** $ACCEPT(T', T, p)$ is true **then**
14: $T = T'$
15: **end if**
16: **end while**
17: **return** (T)

Fig. 6. Method of performing the Perturb operation.

In Fig. 6, the *while* loop iteratively executes the rip-up and re-route operation on the nets. The Perturb should be able to explore diverse solutions for each net. The proposed global router uses only MRF method of routing and generating unique solutions from a single routing method is a difficult task. The repetitive application of the MRF method on a same net in different iterations should produce unique spanning trees for that net, otherwise, there is no advantage of ripping-up and re-routing that net. The MRF method can generate unique spanning trees for a net under the following two conditions: (a) each application has different parameter values, and (b) the usage of the edges is different in each application. The Perturb function cannot alter the parameters of the MRF method because parameter values are controlled by another component of the StocE algorithm but it can alter the usage of edges in each application of the MRF method. It can alter the usage of edges by ripping-up two or more nets. It first rip-up two nets and then re-route them one after another. For each $n_i \in SS$, the proposed Perturb function tries to find another net n_j in *SS* whose bounding box (or routing region) intersects with the bounding box of n_i. The bounding box refers to the same meaning and purpose as described in Section 4.1.1 (MRF Method). When two nets that share a common portion in their bounding boxes are ripped-up then at the time of their re-routing the usage of edges in their bounding boxes would be different from the last time when they were routed. The nets that do have overlapped bounding boxes will be ripped-up and re-routed individually and rely on the assumption that changes in the spanning trees of some nets could have produced changes in the usage of edges.

The *ACCEPT*() function is used to allow hill-climbing and accepts good solutions. Fig. 7 shows the *ACCEPT*() function. It performs the task of accepting or rejecting a new solution which is created by the Perturb operation. As the pseudo-code in Fig. 7 shows that the new solution (T') is compared with the existing solution (T) and is accepted under the following conditions: (i) if $tof(T') < tof(T)$; (ii) $tof(T) - tof(T') \geq -p$ (iii) $tof(T) = tof(T')$, but $tot(T') < tot(T)$; and (iv) $tof(T) = tof(T')$, and $tot(T') = tot(T)$, but $full(T') < full(T)$. The parameter *p* is used for hill-climbing and to avoid getting trapped in local minima. However, it is found in

Input: $G(V, E)$, T: Actual solution, T': Solution on which Perturb operation is applied, $p \in Z^+$
Output: $Y \in true, false$
1: $Y = false$
2: r_{NEG}= a random integer between $-p$ and 0
3: **if** $tof(T') < tof(T)$ or $(tof(T) - tof(T')) \geq r_{NEG}$ **then**
4: $Y = true$
5: **else**
6: **if** $tof(T') = tof(T)$ and $tot(T') < tot(T)$ **then**
7: $Y = true$
8: **else**
9: **if** $tof(T') = tof(T)$ and $tot(T') = tot(T)$ and $full(T') < full(T)$ **then**
10: $Y = true$
11: **end if**
12: **end if**
13: **end if**
14: **return** Y

Fig. 7. Method to accept or reject a solution Y = ACCEPT(T, T', p).

experiments that a very large value of p could waste time by accepting unnecessary bad moves.

4.2.3. Update parameter values

The standard StocE algorithm has three parameters: R, ρ and p which are used to control the optimization loop. This work uses three additional parameters (β, *BOX_SIZE*, and *S_OFL*). The parameters β and *BOX_SIZE* are used in the MRF method. The parameter *S_OFL* mentions a small threshold value of the total overflow such that when the total overflow becomes smaller than that then *BOX_SIZE* becomes equal to its maximum value. The values of some parameters (i.e., p, β, and *BOX_SIZE*) change following an arithmetic progression (AP). An AP can be defined by three terms: (first term, last term and difference). The values of all parameters are initially set equal to their first terms and in the step of update parameter values, their values incremented by the amount equal to their difference values. When the parameters reach their last term, then their next values are equal to their first term. Table 1 lists the parameters and the representation of their APs. The parameter *BOX_SIZE* uses its value BOX_SIZE_i in the initial routing of nets and uses its values of the AP in the R&R process.

Table 2 lists the conditions under which the parameters update their values. The term tof_i refers to the overflow of the current iteration (i.e., i) and tof_{i-1} refers to the overflow of the previous iteration (i.e., $i-1$). The second condition shows that when the overflow becomes smaller than a threshold value (*S_OFL*), then the *BOX_SIZE* size becomes equal to its maximum value so as to allows the nets to use maximum routing region in order to find congestion-free spanning trees. The condition of *BOX_SIZE* mentioned in the second row has priority over the condition which is specified in third row.

5. Experimental results

The proposed global router was implemented in C++ on the Linux platform using eclipse IDE. The code was compiled using Intel Compiler for C++. The implementation of the proposed global router is referred as 'Proposed' in this section. The proposed global router does not use multi-threading. The experiments use two different sets of benchmarks: ISPD98 [13,14] and ISPD2008 [15]. The ISPD98 benchmarks contain industrial problems of 2D global routing whose size vary from small to medium. The ISPD2008 benchmarks contain huge size of problems of 3D global routing. The experimental results on the two benchmarks are discussed in the following in separate sub-sections.

Table 1
List of all parameters and their APs.

Parameters	Representation of their APs
p	$(p_o, p_{max}, \Delta p)$
BOX_SIZE	BOX_SIZE_i, $(BOX\_SIZE_o, BOX\_SIZE_{max}, \Delta BOX\_SIZE)$
β	$(\beta_o, \beta_{max}, \Delta\beta)$

Table 2
Conditions under which parameters update their values.

Parameter	Condition	New value
p	$tof_i \geq tof_{i-1}$	Next value in the AP
BOX_SIZE	$tof_i < S\_OFL$	BOX_SIZE_{max}
BOX_SIZE	$tof_i \geq tof_{i-1}$	Next value in the AP
β	$tof_i \geq tof_{i-1}$	Next value in the AP

Table 3
The ISPD'98 test problems.

Test problem	# of nets	grid size	Test problem	# of nets	grid size
ibm01	11,507	64 × 64	ibm02	18,429	80 × 64
ibm03	21,621	80 × 64	ibm04	26,163	96 × 64
ibm05	27,777	128 × 64	ibm06	33,354	128 × 64
ibm07	44,394	192 × 64	ibm08	47,944	192 × 64
ibm09	50,393	256 × 64	ibm10	64,227	256 × 64

5.1. ISPD98 benchmarks

Table 3 lists the characteristics of problems in the ISPD98 benchmark [13,14]. The values of parameters of the proposed global router were set as follows: $p = (2, 100, 4)$, $R = 3$, $BOX\_SIZE = \{20, (1, \frac{\max(X_{max}, Y_{max})}{2}), 10)\}$, $\beta = (1, 4, 1)$ and $S\_OFL = 10$. Lee's algorithm used the Eq. (6) in the filling process. The MRF method was implemented using Eq. (6). The parameter values were set based on experimental observations to optimize both runtime and solution quality. The proposed global router and the existing ones were executed on a desktop computer that has Intel Core i5-2500 3.3 GHz processor and 4 GB RAM. An executable binary of Box Router 2.0 [23] was obtained from its authors and used in the experiments. The computer uses Ubuntu Linux operating system. All the routers were executed on the same machine.

The implementation of heuristics use random number generators that use a seed value to generate random numbers. The same seed value always generates a same sequence of numbers. Therefore the performance of heuristics may become dependent upon the seed value. In this work, each test problem has up to 50 trials and the seed value in each trial was set to a different random value. Therefore, the results of the trials show the performance which is independent from any particular seed value. The proposed global router was compared with six recent global routers that are as follows: (i) Box Router 2.0 [23], (ii) BFG-R [24] (also called as FGR), (iii) Sidewinder [25,26], (iv) MaizeRouter [27], (v) Archer [17], (vi) FastRoute 3.0 [28] which uses multi-threading to improve its runtime, (vii) authors' own Game theory based global router (GT-router) [29]. The Box Router 2.0 was executed with default parameters with one change that R&R iterations were allowed in order to minimize the wire-length (i.e., the value of REROUTING_REPEAT was changed from 0 to 100). The comparison does not include any EA-based global router because none of the existing EA-based global router has solved the ISPD98 suite of test problems. The comparison uses two types of results of the existing global routers: (a) the results published in the literature and (ii) the results obtained by executing the existing global router.

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 runtime. Table 4 shows the overflow results produced by the proposed and some recent global routers. The results show that the proposed global router, Box Router 2.0, BFG-R, MaizeRouter, Archer, FastRoute 3.0, and GT-router have solved all problems in the ISPD98 suite, whereas, Sidewinder has solved only seven out of 10 problems. Therefore, the proposed global router is equal to the best ones in terms of its ability to successively remove congestion.

The second important objective of global routing process is to minimize the wire-length because solutions with smaller wire-length have less manufacturing cost. Table 5 reports the results of wire-length of the proposed global router in all trials. The results consist of three columns that are represented as 'Q1', 'median' and 'Q3'. The 'median' column contains the middle value of the all trials. The columns 'Q1' and 'Q3' contain the lower quartile or 25th percentile and upper quartile or 75th percentile of the results in

Table 4
Overflow results on the ISPD98 benchmarks.

Problem	Proposed	Box Router 2.0 [23]	BFG-R [24]	Sidewin,der [25,26]	MaizeRouter [27]	Archer [17]	FastRoute 3.0 [28]	GT-based router [29]
ibm01	0	0	0	255	0	0	0	0
ibm02	0	0	0	2	0	0	0	0
ibm03	0	0	0	0	0	0	0	0
ibm04	0	0	0	618	0	0	0	0
ibm06	0	0	0	0	0	0	0	0
ibm07	0	0	0	0	0	0	0	0
ibm08	0	0	0	0	0	0	0	0
ibm09	0	0	0	0	0	0	0	0
ibm10	0	0	0	0	0	0	0	0

Table 5
Wire-lengths of the proposed global router in all trials.

Problem	Proposed		
	Q1	Median	Q2
ibm01	64,212	64,327	64,450
ibm02	171,665	171,784	171,853
ibm03	146,927	146,991	147,037
ibm04	169,884	169,986	170,104
ibm05	409,764	409,764	409,764
ibm06	279,834	279,919	280,036
ibm07	368,411	368,520	368,664
ibm08	405,752	405,830	405,922
ibm09	414,729	414,789	414,837
ibm10	583,320	583,426	583,686

different trials, respectively. Q1 and Q3 indicate the minimum and maximum values between which results of most of the trials exist. Table 5 has a small difference between the lower and upper quartile values which indicates that the proposed global router is stable and its results are not dependent on any particular seed value.

Table 6 compares the wire-length (median) of the proposed global router with that of existing global routers using their published results. In column 2, the number enclosed in brackets mentions the number of global routers from whom the wire-length of the proposed global router is smaller. The wire-lengths that are larger than that of the proposed global router are represented in bold numbers. The results of Box Router 2.0 has two columns. The column 'speed' contains the results when its parameters were set for maximum speed and the column 'wire-length' contains results when its parameters were set for minimum wire-length. A comparison of the results convey the following information about the wire-length of the proposed global router: (i) it is always better than that of Box Router 2.0 (speed) and Sidewinder and the average difference is $+1.51\%$ and $+0.53\%$, respectively; (ii) when compared to Box Router 2.0 (wire-length), it is better in one test problem and in the remaining problems the difference is small and the average percentage difference is -0.30%; (iii) it has an average percentage difference of -0.33% with BFG-R which is still a small difference; (iv) when compared to MaizeRouter, its wire-length is better in seven out of ten test problems and the average percentage difference is almost zero; (v) when compared with Archer, its wire-length is smaller in five out of 10 test problems, however, the average percentage difference is only -0.01%; and (vi) when compared with FastRoute 3.0, its wire-length is smaller in up-to six test problems and the average percentage difference is almost zero. In short, the wire-length of the proposed global router is smaller than at-least two global routers in each test problem.

Table 7 compares the wire-length (median) of the proposed global router with the wire-lengths of Box Router 2.0 and GT-router using the results that were obtained from experiments. Table 7 shows the following information about the wire-length of the proposed global router: (i) it is better than that of Box Router 2.0 in six test problems and the average difference is almost zero; and (iii) it is better than that of GT-router is all test problems and the average difference is 1.04%. The experimental results are also used in the runtime comparison which is described in the following text.

The runtime is also considered important in the design of global routers [5]. Table 8 shows the runtime of the proposed global router as well as that of others. The runtime results of the proposed global router has three columns. The first column is labeled as 'Q1' and refers to lower quartile. The second column 'median' refers to median and the third column 'Q3' refers to the lower quartile. The runtime values which are bigger than that of the proposed global router are represented in bold numbers. The results show that the proposed global is faster than Box Router 2.0 in nine out of 10 problems. The results of Tables 7 and 8 show that the proposed global has produced solutions of smaller wire-length as well as has smaller runtime as compared to Box Router 2.0.

A unique feature of the StocE algorithm is its ability to hill-climb in-order to avoid getting stuck at local minima. It was observed in the experiments that StocE algorithm performs hill-climbing in almost all problems in-order to converge to globally optimal solution. Fig. 8 shows two optimization curves. The curve (a) uses hill-climbing, whereas, the curve (b) does not use hill-climbing. The curves show a window of iterations from 7th to 30th. The overflow value in the first iteration of both curves was 862. The curve (a) indicates that StocE's hill climbing has successfully help in minimizing the overflow to zero value in the 27th iteration, whereas, the curve (b) did not converge to zero value even after 100 iterations. The problem used to illustrate the plots was ibm04. The parameters used in the curves were: (a) $p=(2,100,4)$, $R=3$, $BOX\_SIZE=\{20,(1,48,10)\}$, $\beta=(1,4,1)$ and $S\_OFL=10$, and (b) $p=(0,0,0)$, $R=3$, $BOX\_SIZE=\{20,(1,48,10)\}$, $\beta=(1,4,1)$ and $S\_OFL=10$. The curve (b) has p equal to zero because it has no hill-climbing.

Some recommendations on the setting the parameter values have been prepared based on the experimental observations that are as follows. It was observed that the parameters p and S_OFL effect the routability and runtime, whereas, the parameter β effects the wire-length of the solution. The parameter p controls the amount of hill-climbing and avoid trapping into local minimum solution (i.e., a solution whose overflow is greater than zero). The initial value of β should not be kept high, otherwise it could result in longer wire-length. It is also found that S_OFL should not be kept small, as it could prevent reaching a valid solution. An unnecessary large S_OFL value can increase the runtime and wire-length. The initial value of BOX_SIZE should be kept smaller in order to keep the wire-length smaller and its value should be

Table 6
Comparison of the wire-length of the proposed global router with others using their published results.

Problem	Proposed	Box Router 2.0 [23]		BFG-R [24]	Maize Router [27]	Archer [17]	FastRoute 3.0 [28]	Sidewinder [25,26]
		Speed	Wire-length	Wire-length				
ibm01	64,327 (2)	**66,529**	62,659	63,332	63,720	**64,389**	64,221	–
ibm02	171,783 (3)	**180,053**	171,110	168,918	170,342	**171,805**	**172,223**	–
ibm03	146,991 (5)	**151,185**	146,634	146,412	**147,078**	**146,770**	**146,753**	**147,524**
ibm04	169,986 (3)	**176,765**	167,275	167,101	**170,095**	169,977	**170,146**	–
ibm05	409,764 (2)	–	–	409,739	**410,031**	409,761	–	**409,778**
ibm06	279,919 (2)	**288,420**	277,913	277,608	279,566	278,841	279,471	**280,007**
ibm07	368,520 (5)	**377,072**	365,790	366,180	**369,340**	**370,143**	**369,023**	**381,694**
ibm08	405,830 (4)	**418,285**	405,634	404,714	**406,349**	404,530	**405,935**	**413,300**
ibm09	414,789 (4)	**431,298**	413,862	413,053	**415,852**	414,223	**414,913**	**416,554**
ibm10	583,426 (5)	**610,680**	**590,141**	578,795	**585,921**	**583,805**	582,838	**591,036**

Notes: (i) speed indicates that the parameters are set for minimum runtime.
(ii) wire-length indicates that the parameters are set for best solution quality.
(iii) Bold numbers indicate the wire-lengths of the published works that are bigger than the proposed router.

Table 7
Comparison of the wire-length of the proposed global router with others using the results from experiments.

Problem	Proposed	Box Router 2.0 [23]	GT-router [29]
ibm01	64,327	63,531	**66,953**
ibm02	171,783	**172,382**	**175,164**
ibm03	146,991	**147,492**	**149,076**
ibm04	169,986	168,257	**174,419**
ibm05	409,764	**411,138**	**417,756**
ibm06	279,919	279,866	**286,025**
ibm07	368,520	368,002	**373,977**
ibm08	405,830	**406,897**	**414,076**
ibm09	414,789	**417,303**	**421,702**
ibm10	583,426	**587,873**	**593,544**

Table 8
Runtime of the proposed global router and others.

Problem	Proposed (s)			Box Router 2.0 [23] (s)
	Q1	Median	Q2	
ibm01	5.07	5.47	5.83	**7.95**
ibm02	5.60	5.72	5.89	**16.50**
ibm03	4.80	4.89	4.96	**9.98**
ibm04	22.94	30.28	38.79	**37.00**
ibm05	10.85	10.90	10.96	10.08
ibm06	12.50	13.03	13.38	**20.82**
ibm07	13.67	13.87	14.11	**29.27**
ibm08	18.76	19.37	19.90	**35.64**
ibm09	16.66	16.89	17.09	**42.05**
ibm10	26.27	26.64	27.34	**44.98**

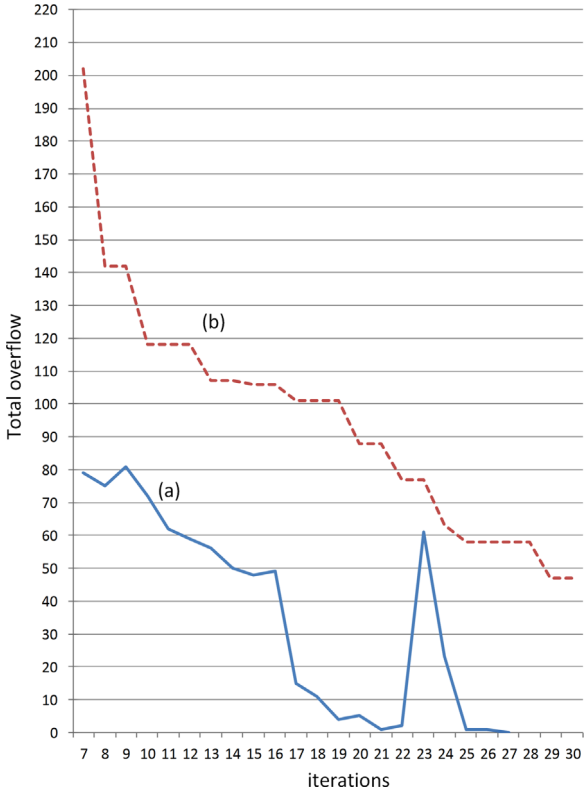

Fig. 8. Illustration of the role of hill-climbing in the StocE algorithm based congestion minimization.

increased in order to route the nets that require very large area, i.e., the maximum value of the AP of the *BOX_SIZE* should be kept large. The initial value of *R* should be set according to the guidelines of the standard StocE algorithm [12].

5.2. ISPD2008 benchmarks

The ISPD2008 suite contains 16 problems [15] of 3D global routing. The proposed global router was evaluated using the first eight problems as most of the existing global routers have been experimented only on the first eight problems. The 3D global routing is usually performed by solving a 2D global routing problem and then using a layer assignment method to convert the results of 2D global routing into 3D global routing. This work presented a 2D global routing method which does not need to perform layer assignment. The experiments showed that it is also efficient in 3D problems whose sizes are many times greater than the 2D problems. Any layer assignment technique can be added into it in-order to convert its results into 3D global routing. Table 9 lists the characteristics of the problems in the ISPD2008 suite.

Table 9
The ISPD 2008 test problems.

Test problem	# of nets	Grid size
adaptec1	219,794	324 × 324
adaptec2	260,159	424 × 424
adaptec3	466,295	774 × 779
adaptec4	515,304	774 × 779
adaptec5	867,441	465 × 468
newblue1	331,663	399 × 399
newblue2	463,213	557 × 463
newblue3	551,667	973 × 1256

The proposed global router used the following parameters: $p=(0,10,100)$, $R=3$, $BOX\_SIZE=\{20,(4,\frac{\max(X_{max},Y_{max})}{2}),10)\}$, $\beta=(0.5,3,0.2)$, $S\_OFL=10$, and $penalty=100$. Eq. (5) was used in the MRF method. The maximum time allowed us to solve any one problem was set as 48 h. The proposed global router was executed on a virtual machine (VM) which has up-to eight 2.1 GHz vCPUs and 16 GB of memory. A vCPU is a simulated CPU that can execute one thread. The VM was configured on the university's cloud computing facilities that consists of a multi-processor system having several Intel Xeon 2.1 GHz processors. The cloud computing server is shared by many users, therefore, the performance of a VM is usually lesser than a dedicated physical machine of the same specifications.

Table 10 shows the results of the total overflow of the proposed global router and three existing global routers (Box Router 2.0 [23], MaizeRouter [27], and Archer [17]). The published results of the global routers were used in comparison. In the ISPD2007 and 2008 global routing contests, runtime has lesser priority as compared to overflow and wire-length, therefore, many existing global routers did not reported the runtime. However, the maximum allowed runtime was chosen as 48 h in Box Router 2.0 [23] (which is same as the proposed global router). The results show that the proposed global router has performed better than the existing global routers. The proposed global router has solved six out of eight problems (which are adaptec1, adaptec2, adaptec3, adaptec4, adaptec5, and newblue2). Among the existing global routers; MaizeRouter has solved five problems and Box Router 2.0 and Archer has solved six problems. The proposed global router has also achieved a smaller overflow value in problems: newblue1 and newblue3 as compared to other global routers.

In 3D global routing, the number of vias also contribute to the wire-length. In ISPD 2008 global routing contest [15], each via contributes one unit of wire-length. However, in the absence of any layer assignment method, the exact 3D wire-length cannot be determined. Therefore, the lower and upper bounds of the wire-length that includes vias was computed in the experiments. Table 11 shows the wire-length results. The results of the proposed global router has three columns: the first column *tot* mentions the segment wire-length which is same as the wire-length computed in 2D problems; the second column contains the lower bound of the wire-length including vias ($\Omega(totv)$); and the third column mentions the upper bound on the wire-length including ($O(totv)$). The lower bound is computed under the assumption that each via connects any two consecutive layers. The upper bound is computed under the assumption that each via connects all layers in the problem. The table shows the wire-lengths of the problems which have been successfully solved by the global routers.

The results in Table 11 shows that the 3D wire-lengths of Box Router 2.0, MaizeRouter and Archer lie between the upper and lower bounds of the 3D wire-length of the proposed global router. Therefore, the 2D wire-length produced by the proposed global router is small and with an efficient layer assignment method, it can produce 3D wire-lengths that will be comparable to the existing global routers. An executable binary of the proposed global router is available at the authors website [30] for Linux 64-bit platforms.

6. Conclusion and future work

This paper proposed a new global router in which the rip-up and re-route (R&R) process is modeled using Stochastic evolution (StocE) algorithm. The existing global routers have complex designs. The main contribution of the proposed global router is that it uses StocE-algorithm to simplify the R&R process. The proposed global router consists of two main steps: (i) generation of initial solution; and (iii) StocE-based R&R process. It uses only maze routing with framing (MRF) method of routing. The initial routing of nets uses Flute-based net decomposition to produce wire-length-efficient initial solution. The StocE algorithm based R&R process eliminates congestion from the initial solution. The StocE algorithm consists of three main steps: (i) Determination of movable elements whose task is to select the nets that are more likely to reduce congestion as compared to others; (ii) Perturb operation whose task is to form a new solution by ripping-up and re-routing the movable elements; and (iii) Update the parameter values whose task is to change the parameter values of routing

Table 10
Overflow results on the ISPD2008 benchmarks.

Problem	Proposed	Box Router 2.0 [23]	MaizeRouter [27]	Archer [17]
adaptec1	0	0	0	0
adaptec2	0	0	0	0
adaptec3	0	0	0	0
adaptec4	0	0	0	0
adaptec5	0	0	2	0
newblue1	360	400	1348	494
newblue2	0	0	0	0
newblue3	15,682	39K	32,588	31,928

Table 11
Wire-length results on the ISPD2008 benchmarks.

Problem	Proposed			Box Router 2.0 [23]	MaizeRouter [27]	Archer [17]
	$tot(\times 10^5)$	$\Omega(totv)(\times 10^5)$	$O(totv)(\times 10^5)$	$totv(\times 10^5)$	$totv(\times 10^5)$	$totv(\times 10^5)$
adaptec1	38.07	51.48	118.58	58.37	62.00	57.66
adaptec2	35.83	48.22	110.14	55.69	57.00	54.62
adaptec3	100.19	128.05	237.36	137.96	138.00	135.18
adaptec4	90.64	123.22	266.25	127.79	128.00	126.32
adaptec5	110.23	145.71	323.13	162.11	–	162.71
newblue2	46.98	62.23	138.48	78.88	80.00	77.94

Notes: (i) *tot* indicates the 2D wire-length which does not include number of vias.
(ii) *totv* indicates the 3D wire-length which includes number of vias.

method and StocE algorithm such as increasing the frame size of MRF method and change amount to hill-climbing of the StocE algorithm. The proposed global router was implemented using C++ on a Linux based machine. It has successfully solved most of the problems in the ISPD98 and ISPD2008 benchmarks. In the future, the methods that can enable multi-threaded global routing will be investigated in order to reduce the runtime of the global router through concurrent routing.

Acknowledgments

Acknowledgments are due to King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia for all support.

References

[1] Thomas Lengauer, Combinatorial Algorithms for Integrated Circuit Layout, John Wiley and Sons, Inc., New York, NY, USA, 1990.
[2] Prashant Saxena, Rupesh S. Shelar, Sachin S. Sapatnekar, Routing Congestion in VLSI Circuits: Estimation and Optimization, Springer Science + Business Media, New York, 2007.
[3] Min Pan, Yue Xu, Xanheng Zhang and Chris Chu, FastRoute, An Efficient and High-Quality Global Router, VLSI Design, 2012, Hindawi Publishing Corporation, VLSI Design, vol. 2012, Article ID 608362, 18 pages, 2012, http://dx.doi.org/10.1155/2012/608362.
[4] Xu He, Tao Huang, Linfu Xiao, Haitong Tian, Evangeline F.Y. Young, Ripple: a robust and effective routability-driven placer, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 32 (10) (2013).
[5] Min Pan, Chris Chu, FastRoute: a step to integrate global routing into placement, in: IEEE/ACM International Conference on Computer-Aided Design, 2006, pp. 464–471.
[6] Yanheng Zhang, Chris Chu, GDRouter: interleaved global routing and detailed routing for ultimate routability, in: 49th ACM/EDAC/IEEE Design Automation Conference, 2012, pp. 597–602.
[7] Sadiq M. Sait, Habib Youssef, Iterative Computer Algorithms with Applications in Engineering, IEEE Computer Society Press, Los Alamitos, CA, 1999.
[8] C. Sechen, A. L. Sangiovanni-Vincentelli, Timberwolf 3.2: a new standard cell placement and global routing package, in: Proceedings of the 23rd Design Automation Conference (DAC), 1986, pp. 432–439.
[9] Young-Long Lin, Yu-Chin Hsu, Fur-Shing Tsai, SILK: a simulated evolution router, IEEE Trans. Comput. Aided Des. 8 (10) (1989).
[10] Ke-Ren Dai, Wen-Hao Kiu, Yih-Lang Li, NCTU-GR: efficient simulated evolution-based rerouting and congestion-relaxed layer assignment on 3-D global routing, IEEE Trans. VLSI Syst. 20 (3) (2012) 459–472.
[11] Youssef G. Saab, Vasant B. Rao, Combinatorial optimization by stochastic evolution, IEEE Trans. Comput. Aided Des. Integr.Circuits Syst. 10 (4) (1991).
[12] Sadiq M. Sait, Habib Youssef, VLSI Physical Design and Automation: Theory and Practice, World Scientific Publishers, Singapore, 1999.
[13] C.J. Alpert, The ISPD98 circuit benchmark suite, in: International Symposium on Physical Design (ISPD), Monterey, CA, 1998.
[14] Modified ISPD '98 Benchmarks [Online] ⟨http://cseweb.ucsd.edu/~kastner/labyrinth_vault/benchmarks/index.html⟩(accessed 08.12.14).
[15] ISPD 2008 Global Routing Contest [Online] ⟨http://archive.sigda.org/ispd2008/contests/ispd08rc.html⟩(accessed 16.10.15).
[16] C. Chu, Y.-C. Wong, FLUTE: fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design, IEEE Trans. Comput. Aided Des. 27 (1) (2008) 70–83.
[17] Muhammet Mustafa Ozdal, Martin D.F. Wong, ARCHER: a history-driven global routing algorithm, in: IEEE/ACM International Conference on Computer-Aided Design, (ICCAD 2007), San Jose, CA, 2007.
[18] Wen-Hao Liu, Wei-Chun Kao, Yih-Lang Li, Kai-Yuan Chao, NCTU-GR 2.0: multithreaded collision-aware global routing with bounded-length maze routing, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 32 (5) (2013) 709–722.
[19] M.M. G-Albarran, A.M. F-P-Cesteros, J.M. S-Perez, A routing strategy based on genetic algorithms, Microelectron. J. 28 (1997) 641–656.
[20] H.-Y. Chen, Y.-W. Chang, Global and detailed routing, in: L.T. Wang, Y.-W. Chang, K.-T. Cheng (Eds.), Electronic Design Automation: Synthesis, Verification, and Testing, Elsevier/Morgan Kaufmann, Burlington, MA, 2009, pp. 687–749.
[21] V. Chvatal, Combinatorial Optimization, IOS Press, 2011.
[22] M. Pan, C. Chu, FastRoute 2.0: a high-quality and efficient global router, in: Asian and South Pacific Design Automation Conference, 2007, pp. 250–255.
[23] Minsik Cho, Katrina Lu, Kun Yuan, David Z. Pan, BoxRouter 2.0: a hybrid and robust global router with layer assignment for routability, ACM Trans. Des. Autom. Electron. Syst. 14 (2) (2009).
[24] J. Hu, J.A. Roy, I.L. Markov, Completing high-quality global routes, in: Proceedings of the 19th International Symposium on Physical Design (ISPD), 2010, pp. 35–41.
[25] J. Hu, J.A. Roy, I.L. Markov, Sidewinder–a scalable ILP-based Router, in: Proceedings of the International Workshop on System Level Interconnect Prediction (SLIP), ACM Press, Newcastle, United Kingdom, 2008, pp. 73–80.
[26] J. Hu, High-performance global routing for trillion-gate systems-on-chips (Ph. D. thesis), Computer Science and Engineering, University of Michigan, 2013.
[27] Michael D. Moffitt, MaizeRouter: engineering an effective global router, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 27 (11) (2008) 2017–2026.
[28] Y. Zhang, Y. Xu, C. Chu, FastRoute 3.0: a fast and high quality global router based on virtual capacity, in: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'08), 2008, pp. 344–349.
[29] U.F. Siddiqi, S.M. Sait, Y. Shiraishi, A game theory-based heuristic for the two-dimensional VLSI global routing problem, J. Circuits Syst. Comput. 24 (6) (2005) 1550082-1–1550082-19.
[30] ⟨https://sites.google.com/site/stocerouter/⟩(posted on June 15, 2015).

Sadiq M. Sait obtained his Bachelor's degree in Electronics Engineering from Bangalore University in 1981, and Master's and Ph.D. degrees in Electrical Engineering from KFUPM in 1983 and 1987, respectively. In 1981 Sait received the best Electronic Engineer award from the Indian Institute of Electrical Engineers, Bangalore (where he was born). Sait has authored over 200 research papers, contributed chapters to technical books, and lectured in over 25 countries. Sadiq M. Sait is also the principle author of two books. He is currently Professor of Computer Engineering and the Director of the Center for Communications and IT Research of KFUPM.

Umair F. Siddiqi received his B.E. degree in electrical engineering from NED University of Engineering and Technology, Karachi, in 2002, his M.Sc. degree in computer engineering from King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, in 2007, and the Doctor of Engineering degree from Gunma University, Maebashi, Japan, in 2013. He is currently a research engineer at the Center of Communications and IT Research of Research Institute of KFUPM. His main areas of research interest are electronic design automation (EDA) software, evolutionary algorithms, and AI. He has published over 20 research papers in international journals and conferences.