Chapter 8
SFB-ACO for Submicron VLSI Routing Optimization with Timing Constraints

The arrival of submicron era has created a huge difference on VLSI (very large scale integration): delay on interconnects has far exceeded that on gates so the total delay for a sink can no longer be simply assessed by the length of weighted edges which makes its routing more complicated than ever. Traditional methods for VLSI routing are either infeasible or with a low precision. In this chapter multiple objectives are comprehensively reflected as a cost the optimization problem has been abstracted as constructing a minimal rectilinear Steiner tree with rectangular obstacles (MRSTRO) under timing constraints. Then the relationship between cost sink delay is cautiously discussed partially proved to be contradictory using Elmore delay model which is of high fidelity. To effectively address the MRSTRO problem a synergy feedback based ant colony algorithm (SFB-ACO) is configuredimplemented. In SFB-ACO a synergy function is designed to lead each branch to join others thus reducing the total tree length. Additionally according to the intrinsic contradiction between objective constraint a constraint-oriented feedback module is introduced with the purpose of preventing over-constrain while regulating the formation of solutions. With configuration principle two modules are uniformly connected with existing ACO operators to form a hybridization of deterministic strategies evolutionary process. The experimental results have verified the advantage of SFB-ACO compared to other algorithms or practices on VLSI global routing.

8.1 Introduction

Global routing [1–4] is to arrange each part of the net into different wiring channels and determine the connection of nodes and their initial wiring courses while satisfying certain design requirements. Its result could have a significant impact on the success of follow-up detailed routing and the overall performance of

© Springer International Publishing Switzerland 2015
F. Tao et al., *Configurable Intelligent Optimization Algorithm*,
Springer Series in Advanced Manufacturing,
DOI: 10.1007/978-3-319-08840-2_8

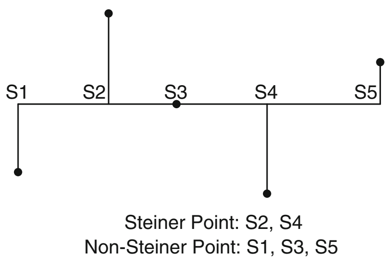


Fig. 8.1 Steiner points and non-Steiner points

the chip [5, 6]. Therefore, it is a core link in very large scale integration (VLSI) physical design. Wires in VLSI can be divided into several types: *signal line*, *power line*, *ground line*, *clock line*, etc., with various optimization objectives of each type. Generally speaking, the total length of interconnect, and the area of wiring district are required to be as small as possible, and time delay for signal lines, clock skew for clock lines are also needed to be considered. Other extra objectives include: power dissipation and heat loss should be reduced; noise and crosstalk between lines should be avoided, etc. These objectives can be comprehensively reflected by weighing each edge in the net; then the optimization problem is narrowed down on minimizing the weighted length of wires, which is usually called a cost. The procedure for constructing interconnects for VLSI global routing has been abstracted as a minimal rectilinear Steiner tree with rectangular obstacles (MRSTRO) problem.

Studies on Steiner tree in Graph theory can be traced to 1941, when Courant and Robbins [7] pointed out that for a net consisting of n endpoints, at most n-2 points are needed to be introduced, and together with the original points, the cost of Steiner tree established on them can be reduced to the lowest. These introduced points are called Steiner points. (Note that not all yielding points are Steiner points, as illustrated in Fig. 8.1) Apparently, how to pick them correctly is a key issue of Minimal Steiner tree (MST) problem [4, 8–10], which has been proved to be a NP-hard problem [4, 9, 11]. The computational amount of some exact algorithms [9, 10, 12, 13] for Steiner tree increases exponentially as the number of nodes increases. Particularly, wires in VLSI must follow Manhattan routing architecture [14, 15], in which only two perpendicular wiring directions are allowed. The optimal tree spanned under such architecture is formatted as a minimal Rectilinear Steiner tree (MRST). Algorithms to perform MRST construction are usually computational-expensive in space, since it requires divisions between each pair of nodes and separated considerations on weighted cost of divided sections, thus a memory usage of prohibitively huge size.

To overcome these shortcomings, Hanan [16] defined Hanan points and gave out a well-known theorem that for any endpoints collection P, there exists a minimum Steiner tree solution, whose Steiner points set S is a subset of its Hanan points set U. Later Snyder [13] demonstrated that the above theorem can be extended to Manhattan space with higher dimensions. Such endeavors greatly make MRST a less overwhelming task. In addition, attentive scholars found that

MST can serve as a good estimation to MRST, and tried to get an approximated solution by using MST as a starting point in efforts to either decompose nets into several two-pin subnets to ease maze routing [3, 17, 18], or simply adopt a pattern technique to restrict the connecting to be L-shape or Z-shape [3, 19, 20]. Hwang [21], in 1976, stated and proved the formula that $cost(MST)/cost(MRST) \leq 3/2$, and this number cannot be improved, which means the approximation cost of MST and MRST will reach 3/2 in the worst cases. Such a poor precision cannot be accepted in VLSI routing design. Meanwhile, the existence of obstacles, which is created by various macro modules, IP modules and some others on the chip, results in not only a more complicated process for deriving MRST from MST, but also a larger discrepancy between their costs. All above have led to an increasing popularity in finding MRST in a more straight and accurate way.

Thus far, numerous mature algorithms directed at MRST have been put forward, such as Geo Steiner package [12, 22], edge based heuristic algorithm [23], 1-Steiner heuristic [24–26] etc. Recently, FLUTE [27, 28] propositioned with improved performance for it is optimal for nets up to degree 9 and is still very accurate for nets up to degree 100. It has been well appreciated and its direct applications are BoxRouter [3, 29] and FastRouter [3, 30, 31]. For tackling MRSTRO problems [32–36] constructed an obstacle-avoiding Steiner tree for an arbitrary λ-geometry by Delaunay triangulation, and demonstrated that it outperformed the conventional construction-by-correction approach [35]. Most algorithms mentioned above are heuristic, which facilitate the finding of a near-optimal solution within a relatively short period of time, thereby has been widely used in multicast network optimization [4]. However, most nets in VLSI circuits have a low degree [28], so rather than having a low runtime complexity, the quality of solution is a more important factor. Discouragingly, up till now, none of the heuristic algorithms can attain twice better performance in the best cases than in the worst ones. Rita and Bryant successfully applied genetic algorithm (GA) in MRSTRO based on MST [36], and Consoli [37] proposed a Jumping Particle Swarm Optimization methodology for addressing the minimum labelling Steiner tree problem, both of which imply a bright and prospective application of intelligent algorithms in VLSI routing [38].

Among intelligent algorithms, ant colony optimization (ACO) [39–41] is a kind of bionic algorithm suggested and quickly developed by Dorigo. Using pheromone to transmit messages between ants, its biggest characteristic is to subtly integrate information of historical experience, excellent solutions and their interactions in a distributed way through weighted edges in the searching space. By receiving positive feedback of pheromone as well as heuristic guidance, the searching and message exchanging efficiency and its quality can be guaranteed, thus gradually becoming a very promising algorithm. At present, ACO is still at the very outset of its development, and is mainly used in path planning and has received better results than genetic algorithm (GA) and simulated annealing algorithm (SA). MRSTRO belongs to path planning, whose optimal route can be excavated relying on distributed information of edges. However, it distinguishes itself from general path planning for it usually contains multiple endpoints. Researches on how to apply ACO on multi-terminal connection is still rare.

On the other hand, integrated circuit develops towards a high-speed and high-integration-level trend. With the coming of deep submicron times, interconnect on VLSI has become thinner and longer, leading to a substantial increase on its resistance and capacitance. Consequently, the delay on interconnect is no longer negligible, while that on gate decreases as its feature size shrinks. For instance, for 100 nm, the intrinsic switching delay of a MOSFET is 5 ps, whereas the RC response time for 1 mm of interconnect is 30 ps. At 35 nm, this 6-to-1 differential turns into a 100-to-1 difference [42]. These changes have made interconnect routing on VLSI very different from before [43]. Previous Linear delay estimator being the Manhattan distance between nodes, which is more commonsense-based, pales in fidelity compared to Elmore delay [42, 44–47], in which a routing tree with shortest length, though possessing a comparative small wiring area, and sometimes a relatively better synchronicity in critical sinks, pays at other prices. By maximizing sharing of tree's branches, it adds extra nodes to the mainstream from source to sinks and this could severely lengthen the delay. As a result, the delay constraints at some sinks may be violated, which adversely affects the performance of circuits or even leaving it malfunctioning. Since the calculation of each sink's delay depends on the structure of Steiner tree and is highly coupled with other branches, such information is rather difficult to be incorporated into distributed edges and hence cannot be appraised by tree's cost. Therefore, traditional approach, to empirically identify delay as connecting length, and then focus objectives of global routing on reducing the total cost, is not applicable in today's deep submicron regime.

Based on the above analysis, the delay on each sink is intrinsically contradicted to the total wiring length. In other words, to ensure the least delay on a particular sink, what we need to do is just to link it directly with the source, which may lead to a star-like topology of Steiner tree. Obviously, its total cost is relatively high and thus unwanted. When the delay calculated from the resulting tree is less than the given constraint, it usually means that there is still possibility for merging branches to reduce cost. Therefore, the ideal situation is that delay on each sink should be less than but as close as possible to their respective constraints. Normal methods to deal with optimization problems with constraints can be roughly summarized as follows: one is to accept or abandon a solution (AAS) directly in relation to its eligibility to meet the constraints, and the other one is to bring in a penalty function to turn the questions into non-constraint ones. The former one fails to make full use of solutions with good target values but cannot satisfy the constraints, and in the latter one, likewise, such solutions suffer from punishment and then degrade. In this context, we hope to take advantage of delay information in the last iteration as guidance for generating solutions in the next. Inspired by the positive feedback in ACO, and considering the contradictory relationship between objectives and constraints in VLSI global routing, a negative feedback is introduced to reconcile merging of branches according to the degree how a constraint is satisfied. Specifically, if the delay constraint on a sink is severely violated, its synergy coefficient decreases so as to restrain meeting with others, and otherwise increases to encourage so.

The primary works of this chapter are as follows:

Optimization on global routing in VLSI is abstracted as a MRSTRO problem. For addressing this problem, the MST constructing process is skipped, and an enhanced ACO, which contains a synergy function other than the pheromone and heuristic factors, is proposed and applied in multi-terminal path planning problems.

Differences on VLSI routing between today's deep submicron era and before are carefully investigated. A more accurate Elmore delay is employed and a constraint-oriented feedback is introduced to adjust branch's merging with others to prevent the case of over-constrain.

Through experimental tests, the effectiveness of synergy function and constraint-oriented feedback in our proposed SFB-ACO is verified by comparisons with other algorithms or practices.

8.2 Preliminary

8.2.1 Terminology in Steiner Tree

The model for VLSI global routing in this chapter is based on MRSTRO, where Steiner tree consists of a collection of given points, additional introduced Steiner points and their connecting relationships. For any two points in Steiner tree, one and only one path can be found. Other requirements in VLSI routing include: either horizontal or vertical wiring lines, no transverses across functional area on the chip. Here, some interpretations related to Steiner tree [8, 10] need to be given as follows.

- ***Root, Node, Leaf and Edge***
  Any point consisting of a Steiner tree is called a node, whose set, denoted as T, is a union of P and S. Connection between two nodes is called an edge, which defines a parent-child relationship. Within the structural hierarchy in the tree, there is a node with special status, usually called as a root. The closer to the root, the higher rank of the node is. Leaf is defined as a node whose degree is 1, namely the point that only has one connection.
- ***Steiner Points and Hanan Points***
  Any additionally introduced points that can help reduce the length of the spanning tree are called as Steiner Points. By drawing horizontal and vertical lines through points in P, we can obtain a Hanan grid. The intersections of the grid are called Hanan points, and its collection is indicated as U.
- ***Mainstream, Segment and Subtree***
  For any element in P, the path from it to the root is called its mainstream, and the connection between two adjacent nodes in the mainstream is called a

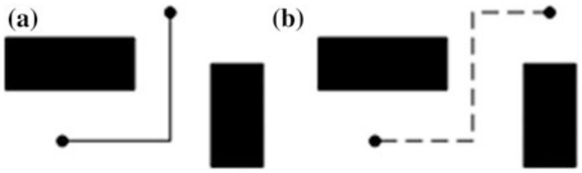


Fig. 8.2 Instances of connectivity **a** connectable **b** non-connectable

segment. A partial tree rooted in node T_i, and consisting of T_i and all its child nodes are called a subtree of T_i, denoted as $Sub(i)$.

- ***Connectivity***
  In the wiring diagram, the existence of obstacles may prevent some Hanan points to be selected as Steiner points. In the cases where exists a pair of nodes P_1 and P_2, and their Hanan points U_1 and U_2, their connection cannot be completed by simply choosing U_1 or U_2 to be the yielding point, but requires two or more, such situation is called non-connectable; otherwise, we call it connectable, illustrated in Fig. 8.2.

8.2.2 Elmore Delay

Elmore delay is a relatively accurate and commonly used model when calculating signal delay over the network. In today's deep submicron era, delay on the VLSI interconnect can no longer be ignored. For a wire with length L, it can be divided into N segments and each is measured as ΔL; then it can be described by a RC network model illustrated in Fig. 8.3.

Assuming that the wire itself is homogeneous, that is, the resistance and capacitance per unit length is a constant, denoted as R_{rate} and C_{rate}, respectively, then the total delay along this wire can be expressed in Eq. (8.1) [45].

$$\begin{aligned}\tau_L &= (R_{rate}\cdot\Delta L)(C_{rate}\cdot\Delta L)+2(R_{rate}\cdot\Delta L)(C_{rate}\cdot\Delta L)+\cdots+N(R_{rate}\cdot\Delta L)(C_{rate}\cdot\Delta L)\\ &= R_{rate}\cdot C_{rate}\cdot(\Delta L)^2\cdot\sum_{i=1}^{N} i=\frac{1}{2}R_L\cdot C_L\end{aligned} \tag{8.1}$$

where, R_L and C_L represent the wire's total resistance and capacitance.

For a node T_i in Steiner tree, the signal delay from the root T_0 to T_i can be formulated as Eq. (8.2) [46].

$$\tau(T_i) = R_{T_0}\cdot C_{T_0} + \sum_{all\,segments\,along\,mainstream(i)} \tau_{e_j} \tag{8.2}$$

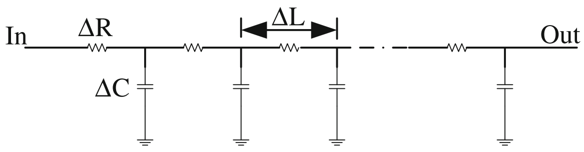


Fig. 8.3 RC model for a wire

$$\tau_{e_j} = R_{T_j} \cdot C_{T_j} + R_{e_j}\left(\frac{C_{e_j}}{2} + C_{sub(j)}\right) \tag{8.3}$$

$$C_{sub(j)} = \sum_{all\,nodes\,in\,subtree(j)} C_{T_k} + \sum_{all\,edges\,along\,subtree(j)} C_{e_l} \tag{8.4}$$

In Eq. (8.2), $R_{T_i}(i = 0, 1, 2, \ldots)$ represents the resistance to drive node T_i, $C_{T_i}(i = 0, 1, 2, \ldots)$ represents capacitance of node T_i, and e_j represents the edge from T_j to its next nearest node along T_i's mainstream. Correspondingly, delay along this edge is denoted as τ_{e_j}, and computed by Eq. (8.3), where R_{e_j} and C_{e_j} respectively represent the total resistance and capacitance of the edge, and $C_{sub(j)}$ is defined as the equivalent capacitance of subtree rooted in T_j's nearest child node along the mainstream, which is the sum of capacitance of all nodes and edges in the subtree. If the child node of T_j is a leaf, then the value of $C_{sub(j)}$ is identical to the capacitance of this leaf.

Up till now, for each node in the Steiner tree, its delay from the root along its mainstream can be calculated iteratively according to Eqs. (8.2)–(8.4).

8.2.3 Problem Formulation

This section discusses MRETRO problem with timing constraints for VLSI global routing in a deep submicron era. In this section, the relationship between the sink delay and tree length is scrutinized, and appropriate candidate pool for Steiner points is determined on account of solution precision and space complexity.

Given a point set $P = \{P_i | i = 1 : n\}$ corresponding to the sinks on the VLSI chip to be optimized, where n is the number of sinks, and root T_0 corresponds to the source on chip. For each sink and source, it can be located by its coordinate (x_i, y_i) on board. Besides, there is a collection of modules, viewed as obstacles, the actual shape of which does not affect the area for wiring because of the Manhattan rule in VLSI, and thus can be formulated or divided into a number of rectangles. These rectangular obstacles are denoted as $R = \{R_i | i = 1 : r\}$, where r is the number of obstacles, and its position and size are expressed by its bottom-left and upper-right vertex coordinates. Upon the basis of sinks and source, some other special points, known as Steiner Points, are needed to be introduced. How to construct a MRST using these points, and at the same time, not violate the delay constraint of each sink? This is an issue which needs to be addressed here.

As mentioned before, the wiring length, chip area, power consumption, heat loss, and clock synchronicity can be accessed by the total cost of weighted edges in the spanning tree. Here all the wires in the chip are assumed to be homogeneous, which means that all edges are of uniform weight (which is set to 1), so that minimizing the total cost of Steiner tree is equivalent to minimizing its total length. Also, the delay constraint of each sink is set to be $T_{limit} = \{T_{limit}(i) | i = 1 : n\}$, then the optimization problem can be formulated as follows.

$$Min\left(\sum_{all\ segments} L_e\right) \tag{8.5}$$

$$subject \quad to \quad \tau(T_i) \le T_{limit}(i), \quad \forall i = 1 : n \tag{8.6}$$

In Eqs. (8.5) and (8.6), L_e represents the length of each segment in the spanning tree, and τ and T_{limit} respectively represent the actual Elmore delay of the sink and its delay constraint.

Apparently, the selection of Steiner points is the key to solve above problem. Note that if rendering the candidate pool to be infinite or with little limitation, it will unavoidably increase the space complexity of the problem.

Theorem 1 (Hanan [16]) *For any MRST, all of its Steiner points are Hanan points.*

Corollary 1 *If a MRSTRO problem is solvable, its optimal solution can be obtained by selecting Steiner points from Hanan points set or from points located in the rim of obstacles.*

Corollary 2 *For two points T_i and T_j to be connected, the shortest path between them is equal to the Manhattan distance between them, as defined in Eq. (8.7), when they are connectable; otherwise it should at least contain one portion of obstacle's edge.*

$$D(T_i, T_j) = |x_i - x_j| + |y_i - y_j| \tag{8.7}$$

Theorem 2 *For a partial tree T and an unconnected point P_k outside the tree, the best location in segment for a Steiner point to connect P_k to T to control the tradeoff between tree length and sink delay, should lie between SP and CUC, as shown in Fig. 8.4, where SP is the shadow point of P_k to the segment, and CUC is the closest upstream connection to P_k.*

Proof Let T_i denote the closer-to-root endpoint of the segment to be connected, which is CUC, and its coordinate to be (0, 0). Let the other endpoint, i.e., closest downstream connection (CDC), to be indicated by T_j, with its coordinate being (x_j ,0). Let the coordinate of P_k to be (x_k, y), and its shadow point in segment to be $(x_k, 0)$. Let R_{T_i} and C_{T_i} denote the resistance and capacitance of node T_i, respectively. Let R_s and C_s respectively represent the Steiner point's resistance and capacitance if it does not lie on CUC or CDC. Denote the resistance and

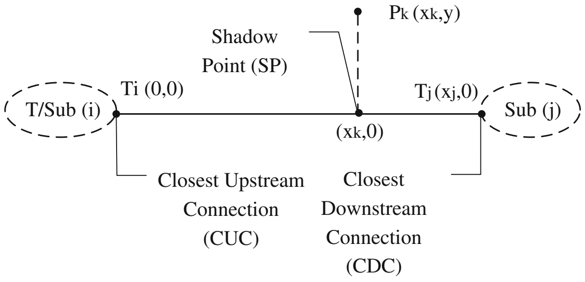

Pk (xk,y)
Shadow Point (SP)
Ti (0,0)
T/Sub (i)
Tj (xj,0)
Sub (j)
(xk,0)
Closest Upstream Connection (CUC)
Closest Downstream Connection (CDC)

Fig. 8.4 Diagram of CUC, SP and CDC

capacitance per unit length on segment by R_{rate} and C_{rate}, respectively, and the equivalent capacitance of subtree rooted in T_j to be $C_{sub(j)}$, and assume that the coordinate for selected Steiner point on segment is $(x,\ 0)$. Then according to Elmore model, delay from source to node P_k along its mainstream can be expressed as follows.

$$\tau(P_k) = \tau(predecessor) + \tau_{e_i} + \tau_{e_S} \tag{8.8}$$

In Eq. (8.8), $\tau(predecessor)$ represents the signal delay from source to node T_i, τ_{e_i} represents delay from node T_i to the selected Steiner point, and τ_{e_s} represents delay from the Steiner point to node P_k.

Additionally, we can easily infer from Eqs. (8.2)–(8.4) that the connecting of P_k only affects the value of C_{sub}, and for each $C_{sub(i)}$ in the upstream route, it can be expressed as in Eq. (8.9).

$$C_{sub(l)} = C_{predecessor} + C_{sub(i)}, \quad \forall T_l \in Predecessor(T_i) \tag{8.9}$$

So that we can rewrite Elmore delay to be a linear function of $C_{sub(i)}$, expressed as follows.

$$\tau(predecessor) = ConA + ConB \cdot C_{sub(i)} \tag{8.10}$$

Where *ConA* and *ConB* are both constant, which are only related to the resistance and capacitance of T_i's upstream route, respectively. The change of $C_{sub(i)}$ has nothing to do with the value of *ConA* or *ConB*.

Hence the influence of Steiner point's location on $C_{sub(i)}$ can be calculated according to Eq. (8.11).

$$C_{sub(i)} = C_{T_i} + C_S + C_{P_k} + C_{sub(j)} + C_{rate} \cdot \left(x_j + |x_k - x| + y\right) \tag{8.11}$$

Also, its influence on τ_{e_i} and τ_{e_s} can be expressed as in Eqs. (8.12) and (8.13), respectively.

$$\tau_{e_i} = R_{T_i} \cdot C_{T_i} + R_{rate} \cdot x \cdot \left\{ C_{rate} \cdot \frac{x}{2} + C_S + C_{sub(j)} + C_{P_k} + C_{rate}\left[(x_j - x) + |x_k - x| + y\right] \right\} \quad (8.12)$$

$$\tau_{e_S} = R_S \cdot C_S + R_{rate} \cdot (|x_k - x| + y) \cdot \left(C_{rate} \cdot \frac{|x_k - x| + y}{2} + C_{P_k} \right) \quad (8.13)$$

According to Eqs. (8.8)–(8.13), we can easily find that $\tau(P_k)$ is a piecewise quadratic function of x, expressed as follows.

$$\tau(P_k) = A + Bx + Cx^2 \quad (8.14)$$

where

$$A = \begin{cases} ConA + ConB\left[C_{T_i} + C_S + C_{P_k} + C_{sub(j)} + C_{rate}\left(x_j + x_k + y\right)\right] + R_{T_i}C_{T_i} \\ \qquad + R_S C_S + R_{rate}(y + x_k)\left(C_{rate} \cdot \frac{y + x_k}{2} + C_{P_k}\right), \quad 0 \le x \le x_k \\ ConA + ConB\left[C_{T_i} + C_S + C_{P_k} + C_{sub(j)} + C_{rate}\left(x_j - x_k + y\right)\right] + R_{T_i}C_{T_i} \\ \qquad + R_S C_S + R_{rate}(y - x_k)\left(C_{rate} \cdot \frac{y - x_k}{2} + C_{P_k}\right), \quad x_k < x < x_j \end{cases} \quad (8.15)$$

$$B = \begin{cases} -ConB \cdot C_{rate} + R_{rate}\left(C_S + C_{sub(j)} + C_{rate} \cdot x_j\right), & 0 \le x \le x_k \\ ConB \cdot C_{rate} + R_{rate}\left(C_S + C_{sub(j)} + C_{rate} \cdot x_j\right), & x_k < x < x_j \end{cases} \quad (8.16)$$

$$C = \begin{cases} -R_{rate} \cdot C_{rate}, & 0 \le x \le x_k \\ R_{rate} \cdot C_{rate}, & x_k < x < x_j \end{cases} \quad (8.17)$$

Other nodes, apart from those which directly connect to the source without any other nodes within the mainstream, their signal delay will also be affected due to the newly connected point P_k. Among them, nodes located on $T/Sub(i)$ mainly suffer from the change of $C_{sub(i)}$, and the major cause for delay variation of those on $Sub(j)$ will be the increase of segment number and their corresponding delay change along the mainstream. Delay on these nodes is also a piecewise quadratic function of x, which can be get as above.

Qualitatively drawing curves to depict signal delay of P_k and nodes distributed in other positions, as shown in Fig. 8.5, we can easily tell that delay on all sinks increase when the Steiner point is inserted after *SP*. And apparently, the tree length is longer compared to the situation when the Steiner point lies before *SP*. Therefore, an appropriate location for the new Steiner point should be between *CUC* and *SP*. In this region, the length of spanning tree gradually decreases when slowly shifting Steiner point backwards, and reaches its lowest point when at *SP*. Another conclusion drawn from Fig. 8.5 is that, delay on each sink continuously changes as the Steiner point moves between *CUC* and *SP*, and some of them

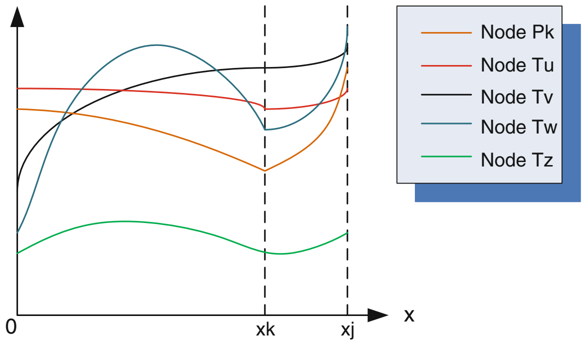


Fig. 8.5 Change of signal delay on different nodes

change in the opposite direction, which implies, that the timing conditions at different sinks are sometimes contradictory when adjusting position of one Steiner point. This makes it possible for us to artificially regulating the synergy function of branches in order to meet their respective timing constraints.

From above, we also see that limiting the candidate Steiner points to the Hanan pool is actually not conductive to the adjustment of sink's delay. And according to Corollary 1 and 2, the Hanan points are not enough if the design model is non-connectable itself. Taking the space complexity into account, $S = U_{refined} \cup RIM \cup PEAK$ can serve as an appropriate candidate pool for Steiner points, where $U_{refined}$ is a collection consisting of Hanan points that lies off the obstacle region, *RIM* is comprised of intersections created by drawing horizontal and vertical lines from sinks to the obstacles' rims, and *PEAK* represents the collection of all rectangular obstacles' vertices, as illustrated in the right panel of Fig. 8.8. □

8.3 SFB-ACO for Addressing MSTRO Problem

This section starts with a brief overview of ACO on two-endpoint path planning, and based on it, a SFB-ACO algorithm encompassing a synergy function and constraint-oriented feedback is proposed and then applied on multi-terminal routing optimization presented before.

8.3.1 ACO for Path Planning with Two Endpoints

The basic idea for traditional ACO can be summarized as below. Using ants' paths to represent feasible solutions, all paths searched can constitute the whole solution space for the given optimization problem. Let ants release more pheromone on the path whose total length is relatively shorter, and as time goes by, more and more pheromone can be accumulated on such paths, and thus they are more likely to be selected by other ants. Influenced by such an intense positive feedback, ants will

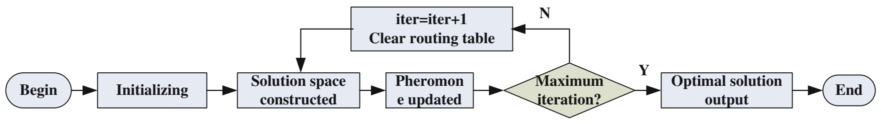


Fig. 8.6 Framework of standard ACO

ultimately converge into an optimal path with shortest length, and this path is also called as the optimal solution.

The main framework for standard ACO on path planning with two-endpoints is depicted in Fig. 8.6.

At first, pheromone concentration on all edges is the same, denoted as $\zeta_{ij}(0) = \tau_0$. Ant k ($k = 1,2,\ldots,m$) will choose the next node to visit according to the amount of pheromone deposited on edge as well as heuristic information, and the corresponding transferring rate for ant k to move from node i to node j can be denoted as $P_{ij}^k(t)$, and calculated in Eq. (8.18).

$$P_{ij}^k = \begin{cases} \dfrac{(\zeta_{ij})^{\alpha} \cdot (\eta_{ij})^{\beta}}{\sum\limits_{S \in allow_k} (\zeta_{ij})^{\alpha} \cdot (\eta_{ij})^{\beta}}, & S \in allow_k \\ 0, & S \notin allow_k \end{cases} \tag{8.18}$$

where ζ_{ij} represents the pheromone concentration on edge between i and j, η_{ij} represents heuristic function to signify expectation for ants to move from i to j, $allow_k$ represents the node collection that are allowed for ant k to visit, α is a pheromone factor, whose value represents the importance degree of pheromone concentration in ant's transferring and value of β, referred to as heuristic factor, represents that degree of heuristic information.

At the same time, the pheromone concentration on each edge will be updated with its formulation expressed as below.

$$\zeta_{ij}(t+1) = (1-\rho) \cdot \zeta_{ij}(t) + \Delta\zeta_{ij} \tag{8.19}$$

$$\Delta\zeta_{ij} = \sum_{k=1}^{n} \Delta\zeta_{ij}^k \tag{8.20}$$

where ρ represents the degree of pheromone evaporation, and its value lies on region [0,1], $\Delta\zeta_{ij}^k$ represents pheromone released by ant k on edge connected from i to j, and $\Delta\zeta_{ij}$ represents the total pheromone released by all ants on this edge.

As for the updating mechanism, Dorigo has given three different models, which are ant cycle system, ant quantity system, and ant density system. Among them, the first one employs the global information on ants' routing, thus being most commonly used. Its updating mechanism is introduced as follows.

$$\Delta\zeta_{ij}^{k} = \begin{cases} \frac{Q}{L_k}, & \mathit{if\ ant(k)\ visit\ node(j)\ from\ node(i)} \\ 0, & \mathit{otherwise} \end{cases} \tag{8.21}$$

where Q is a constant representing the total amount of pheromone released in one cycle, and L_k is identical to the length of ant k's route.

Standard ACO is perfectly suitable for shortest path planning with a sole source and destination. However, in a VLSI circuit board, there are multiple sinks and one source; what requires to be optimized is not the separate path from each sink to the source as in standard ACO, but the whole spanning tree created by all these points. In other words, no branch is completely independent, and only by merging branches in the maximum degree can we expect the shortest length of the tree. That's the reason that a synergy function is introduced in our proposed SFB-ACO.

8.3.2 Procedure for Constructing Steiner Tree Using SFB-ACO

Here we incorporate a synergy matrix γ with size $n \times n$, where $\gamma(i,j)$ represents the function for branch i to join in branch j. The procedure for constructing a Steiner tree is described as in Fig. 8.7. For n sinks to be connected in VLSI, let the number of ants in one group to be n, and that of ant groups to be m. Let the initial positions for n ants in one group to be the places where sink 1, sink 2, …, sink n lie. S is the candidate pool for Steiner points, and in combination with the source and sinks, they make up a point collection for ants to visit. Similar to standard ACO, each ant chooses its next node according to the transferring rate, and creates its own routing table. If any ant in the group transfers to the source or to nodes that have previously been visited by the other ants in its group, it succeeds in finishing its task and its travelling ends. Upon all ants in the group finish their tasks, we check the route to see whether it is a Steiner tree and record its length if yes. Otherwise, if any ant encounters a dead corner, i.e., there are not any allowed nodes to choose, we label a failure on the group and cancel all movements of its ants. Hereto, we call it one cycle. The pheromone concentration is updated once in a cycle, and only the group who successfully finishes the task can release pheromone on its path. As the pheromone accumulates through several cycles, the optimal Steiner tree can be found.

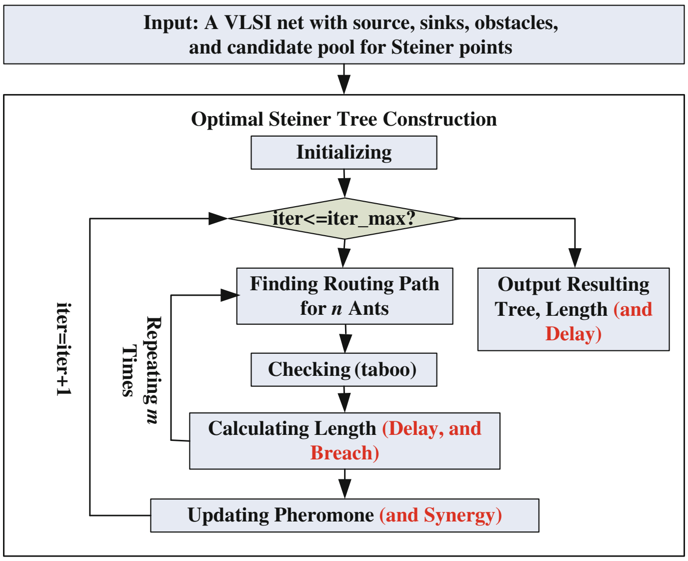

Input: A VLSI net with source, sinks, obstacles, and candidate pool for Steiner points
Optimal Steiner Tree Construction
Initializing
iter<=iter_max?
Finding Routing Path for n Ants
Output Resulting Tree, Length (and Delay)
Checking (taboo)
Calculating Length (Delay, and Breach)
Updating Pheromone (and Synergy)
Repeating m Times
iter=iter+1

Fig. 8.7 Framework of SFB-ACO

The pseudo code for function *FindRoute* in Fig. 8.7 is as follows. In line 9, the transferring rate is calculated as follows.

```
Function taboo = FindRoute(source, sinks, obstacles, candidate Steiner points)
Step1    Initialization taboo, allowedmember
Step2    While (allowedmember ~= NULL)
Step3        Find nodes nearest to the taboo(end) from four directions
Step4        If (edge between node i and taboo(end) goes across any obstacle)
Step5            Let node = node / i
Step6        End if
Step7        Let node = node / taboo
Step8        If (node ~= NULL)
Step9            Calculate transferring rate
Step10           Select node and update taboo
Step11           Calculate transferring rate
Step12           If (node == root || node is a member of other ant's taboo)
Step13               Let current member be erased from allowedmember
Step14           End if
Step15       Else
Step16           Record failure on this round and break
Step17       End if
Step18   End while
Step19   Output the routing table of n ants
```

$$P_{ij}^{k} = \begin{cases} \dfrac{(\varsigma_{ij})^{\alpha}\cdot(\eta_{ij})^{\beta}\cdot fun_{\gamma_{ij}}^{k}}{\sum\limits_{s\in allow_k}(\varsigma_{ij})^{\alpha}\cdot(\eta_{ij})^{\beta}\cdot fun_{\gamma_{ij}}^{k}}, & S \in allow_k \\ 0, & S \in allow_k \end{cases} \tag{8.22}$$

where

$$\eta_{ij} = 1/D\left(T_0, T_j\right) \tag{8.23}$$

$$fun^k_{\gamma_{ij}} = \prod_{r=1,r\neq k}^{n} \left[\frac{\sum_{l=1}^{N^r_{node}} \frac{1}{D\left(T_j, T_{tabu^r(l)}\right)}}{N^r_{node}}\right]^{\gamma_{kr}} \tag{8.24}$$

In Eq. (8.22), η_{ij} is identical to the Manhattan distance from source to node j, expressed as in Eq. (8.23). $fun^k_{\gamma_{ij}}$ represents the synergy function for ant k to transfer from node i to node j, whose value can be obtained by Eq. (8.24), where $T_{tabu^\tau(l)}$ represents the node in the route table, N^r_{node} represents the number of nodes that have been visited by ant r, and γ_{kr} represents the importance of synergy for branch k to join in branch r.

The pseudo code for function *Checking* in Fig. 8.7 is as follows.

```
Function flag = Checking(taboo)
Step1   Initialize flag to be 1
Step2   If (root is not a member of taboo)
Step3       Let flag = 0
Step4   Else
Step5       For i = 1 to n
Step6           Let path(i) record the path sink i which can go as far as it can toward the root
Step7           If (root is not the end of path(i))
Step8               Let flag = 0 and break
Step9           Else
Step10              If path(i) contains repeating nodes
Step11                  Let flag = 0 and break
Step12              End if
Step13          End if
Step14      End for
Step15  End if
Step16  Output checking result indicated as flag
```

Another innovative practice referring to the dealing with constraints will be presented in the next subsection.

8.3.3 Constraint-Oriented Feedback in SFB-ACO

Directed by the above analysis, Elmore delay on each sink is closely related to the number of Steiner points, their positions along the mainstream, and the connecting topology of other branches. To put it simple, the more meeting with others, the more reduction it may cause on the length of spanning tree, whereas adding delay on relevant sinks. Therefore, the value in the synergy matrix will have direct impact on its corresponding sink's delay. Different from early methods dealing with solutions that break the constraints, which is either abandoning them or punishing them, this chapter presents a constraint-oriented feedback on elements in γ with the purpose to prevent the case of over-constrain.

First, the definition of constraint breach should be clarified.

$$Breach(T_i) = \tau(T_i) - T_{limit}(i) \tag{8.25}$$

Obviously, any positive value in vector *Breach* reveals a violation to the constraint. Otherwise, it may indicate a situation of over-constrain, which means that there is still space for further reducing the tree's length. Therefore, the ideal value in vector *Breach* should be equal to zero. However, this may be rather difficult, since the candidate pool for Steiner points we select is far from infinite. For that reason, the value in *Breach* should be lesser and as close as possible to zero.

The mechanism to regulate γ based on *Breach* is shown in Fig. 8.7 with red marks, in which the procedure of finding route path for n ants is the same as in Sect. 8.3.2, and the calculation of sink delay is based on Elmore model given in Sect. 8.2.

In addition, the pseudo code for pheromone updating function *UpdatePh* is as follows.

```
Function zera = UpdatePh(zeta)
Step1     Initialize Delta_zeta, rho
Step2     For i = 1 to m
Step3         If (vector{$delay_i - 1.5 \times T_{limit}$} contains no positive numbers)
Step4             Calculate decay_zeta
Step5             For each edge in the routing path
Step6                 Let corresponding element in Delta_zeta increase by Q/Length(i) × decay_zeta
Step7             End for
Step8         End if
Step9     End for
Step10    Let zeta = (1 - rho) × zeta + Delta_zeta
Step11    Output resulting matrix zeta
```

Coefficient *decay_zeta* in above pseudo code is calculated as in Eq. (8.26).

$$decay\_zeta = e^{-\lambda\times\left(\sum\limits_{l=1:n,\,\text{and Breach}_i(l)>0} |Breach_i(l)|\right)} \tag{8.26}$$

where λ is a constant to be determined, and *decay_zeta* represents the decaying degree of pheromone accumulated because of violation of constraints.

The pseudo code for synergy regulation function *RegulateSy* is as follows.

```
Function gama = RegulateSy(gama)
Step1   Initialize p1, p2, p3, p4
Step2   For i = 1 to m
Step3       For j = 1 to n
Step4           If $Breach_i(j) > 0$
Step5               Multiply the jth row of gama by $e^{-p1\times Breach_i(j)}$
Step6               Multiply the jth column of gama by $e^{-p2\times Breach_i(j)}$
Step7           Else
Step8               If (the average length increases in this iteration
                        and $Breach_i$ contains no positive numbers)
Step9                   Multiply the jth row of gama by $e^{-p3\times Breach_i(j)}$
Step10                  Multiply the jth column of gama by $e^{-p4\times Breach_i(j)}$
Step11              End if
Step12          End if
Step13      End for
Step14  End for
Step15  Output resulting matrix gama
```

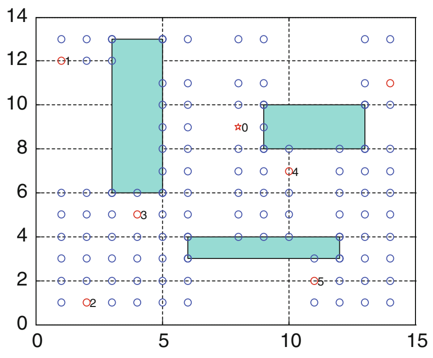
The negative feedback introduced above can effectively direct and regulate the synergy function among branches, thus controlling tradeoff between length and delay. On the other hand, receiving positive feedback from pheromone, paths with desired objective value and can satisfy the constraints will be repeatedly strengthened and strengthened. At last, under the role of double feedback, an optimal solution with its *Breach* value all negative and closest-to-zero can be found.

8.4 Implementation and Results

Experiments have been conducted to evaluate the performance of our proposed SFB-ACO algorithm, and two groups of experiments are designed and carried out. In the first one, based on the same chip consisting of a certain number of sinks and obstacles, two different scale candidate pools for Steiner points are selected; renewed Prim [48], standard ACO, and SFB-ACO are then applied to optimize the routing using the above two pools, assessments with respect to each are made and roles of synergy function and pool size are carefully discussed. In the second experiment, stringent timing constraints are given according to the Elmore delay tested in the first experiment, and a constraint-oriented feedback is introduced in case of over-constrain, and its effectiveness has been validated through comparisons with AAS.

8.4.1 Parameters Selection

In order to apply relevant algorithms on VLSI routing, several parameters have to be determined. Arguments input related to the chip to be optimized, including source, sinks, obstacles and their positions can be graphically obtained from Fig. 8.8, where the above objects are indicated by red star, red circles, and cyan



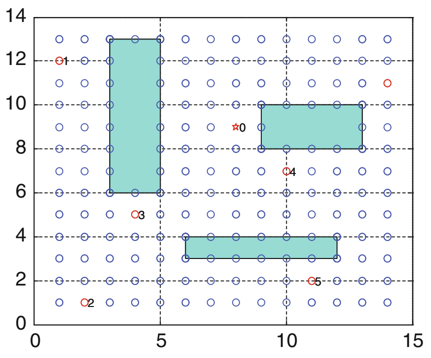


Fig. 8.8 Instance of chip to be optimized and two candidate pools for Steiner points

Table 8.1 Parameters related to algorithms

Parameter	α	β	rho	Q	m	Iter_max	P1	P2	P3	P4
Value	5	20	0.1	2	35	50	0.0000	0.0018	0.0010	0.0000

rectangular. Figure 8.8 also shows two selected candidate pools of different sizes for Steiner points, denoted as pool I and pool II from left to right.

Parameters used in our proposed SFB-ACO are shown in Table 8.1, and those related to the calculation of Elmore delay are summarized in Table 8.2.

8.4.2 Improvement of Synergy

In the first experiment, timing constraints for each sink are set quite loose such that the problem is degraded as a MRSTRO without constraint. Adopting two candidate pools of different sizes for Steiner points, Table 8.3 records the respective results of renewed Prim, standard ACO, and our proposed SFB-ACO, and their optimal routing diagram are given in Figs. 8.9, 8.10, 8.11, 8.12 and 8.13 from top to bottom under two pools. Pool I is a simplified point set of Pool II, in which the points that are not easily accessible, namely, behind obstacles are removed to achieve a lower space complexity. We can see from the data, there is no much difference in the solution quality and convergence rate under two candidate pools. This indeed implies a possibility to reduce algorithm's space complexity while not at the cost of its precision or efficiency. However, this is only valid when leaving the timing constraints aside. If these constraints are stringent, the points behind obstacles may be needed as additional choices for leading a constraint-meet topology of spanning tree. This is the reason that we adopt Pool II in our second experiment.

In Table 8.3, renewed Prim is a kind of greedy algorithm similar to Prim algorithm but with several adjustments mainly considering the Manhattan

Table 8.2 Parameters related to Elmore delay

Parameter	Resistance of				Capacitance of			
	Source (R_{T_0})	Sink (R_{T_i}, $i = 1 : n$)	Node (R_{T_i}, $i > n$)	Edge (R_{rate})	Source (C_{T_0})	Sink (C_{T_i}, $i = 1 : n$)	Node (C_{T_i}, $i > n$)	Edge (C_{rate})
Value	0.4	0.2	0.1	0.1	0.2	0.1	0.05	0.05

Table 8.3 Comparisons of different algorithms under different pools

Algorithm	Minimum length	Average length	Iterations	Elmore delay on each sink					
				Sink 1	Sink 2	Sink 3	Sink 4	Sink 5	Sink 6
Renewed Prim	42	–	–	2.19	1.83	1.65	0.16	2.05	0.32
Adopting pool I for candidate Steiner points (XX elements)									
Standard ACO	43	48	30	1.42	1.44	1.14	0.60	1.44	0.32
SFB-ACO	41	42	12	1.90	1.75	1.53	0.65	1.75	1.03
Adopting pool II for candidate Steiner points (XX elements)									
Standard ACO	42	48	34	1.75	1.66	1.42	0.59	1.45	0.32
SFB-ACO	41	42	18	1.93	1.75	1.51	0.57	1.52	0.32

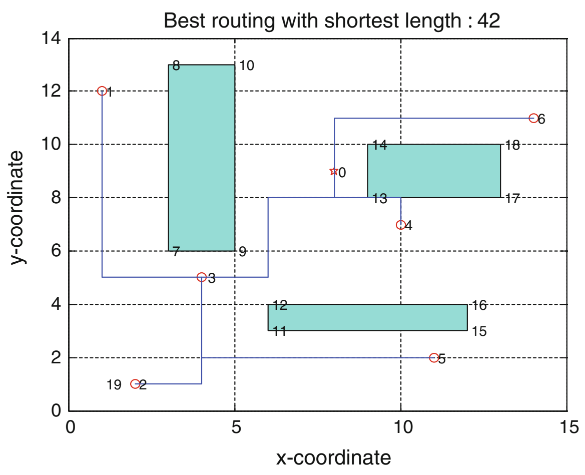


Fig. 8.9 Routing diagram given by renewed Prim

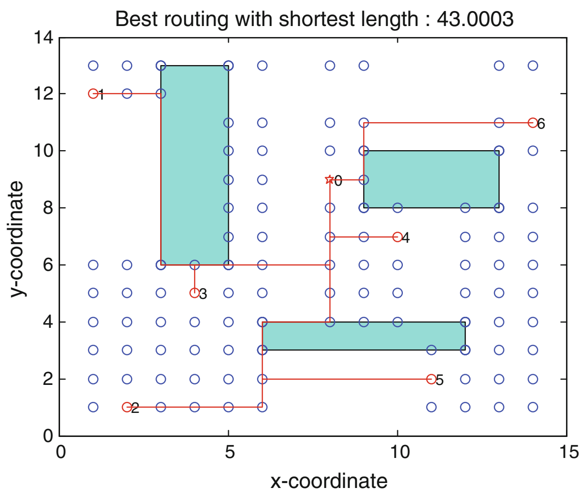


Fig. 8.10 Routing diagram given by standard ACO using pool

architecture and obstacles and its primary mechanism can be described as below. Firstly, starting with a partial tree containing the source, each time we select the sink which has the shortest attainable Manhattan distance to the existing tree. With the selection of sink, the Steiner point can be determined, and therefore an edge between them can be established. The iteration procedure goes on until all sinks have been added to the tree. From above, we know renewed Prim is a relatively deterministic algorithm with quite high efficiency, and that explains why data related to average length and iterations are not recorded in the table. However, in the process of building tree, it is only guided by information given by the added nodes, but without any consideration about the effects it may have on sequential sinks. Its minimum length, 42, though good, is still not the optimal one, compared

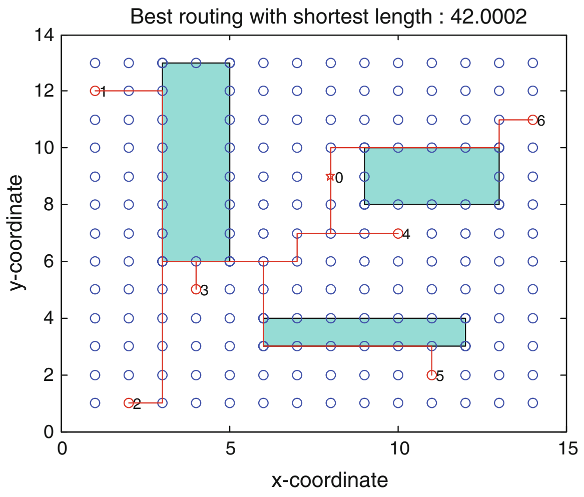


Fig. 8.11 Routing diagram given by standard ACO using pool II

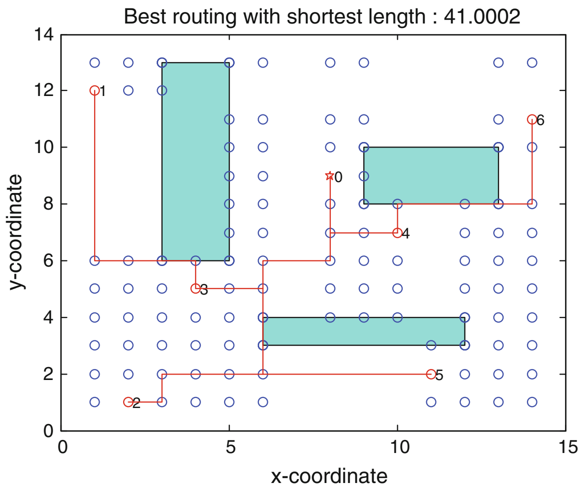


Fig. 8.12 Routing diagram given by SFB-ACO using pool I

to 41 in SFB-ACO. Besides, because of its relative determinacy, it can only obtain solutions with a set of fixed delay on sinks, comparatively, which is absolutely not feasible with stringent timing constraint.

The last two lines in Table 8.3 strongly convince us the advantage of synergy function we introduce in SFB-ACO. Under either pool, SFB-ACO can result in a higher quality of solution and better efficiency of algorithm than standard ACO, 41 versus 43, 41 versus 42, 12 versus 30, and 18 versus 34, respectively. This is because under the function of synergy, branches are no longer independent: they try their best to find ways to join in the tree instead of to reach to the source. Once they merge into another, they just quit travelling and the total length can be

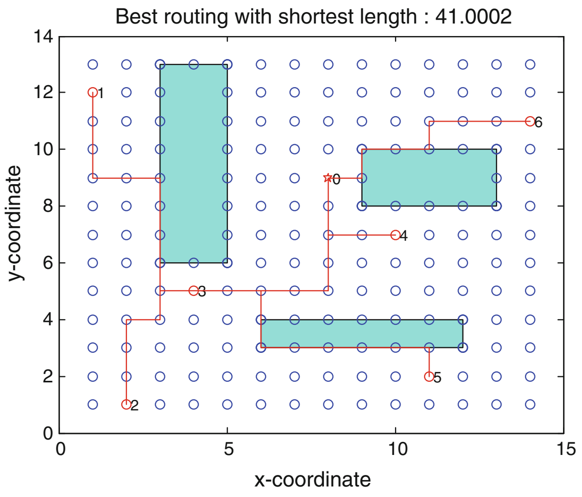


Fig. 8.13 Routing diagram given by SFB-ACO using pool II

reduced. Since the length obtained in their early iterations is already near-to-optimal, the algorithm can converge at a faster rate. Comparing the average length of standard ACO and SFB-ACO, 42 and 48, we also learn that the convergence status in SFB-ACO is better than that in standard ACO. As the iterations goes by, not all solutions can converge into the best one in standard ACO and this is because the pheromone released by the best solution do not have noticeable function on guiding the formatting of its sequential solutions. It, on the other side, implies that merely accounting for pheromone and heuristic information is not enough. Other force, such as our proposed synergy function, is indeed needed.

By comparison of data in Table 8.3 and routing scheme in Figs. 8.9, 8.10, 8.11, 8.12 and 8.13, we also see that same length of two schemes does not necessarily suggest the same topology of spanning tree, not to speak the same delay on each sink. Another purpose for recording Elmore delay in the last couple of columns is for later use as references to giving constraints.

8.4.3 Effectiveness of Constraint-Oriented Feedback

This part will use Pool II for candidate Steiner points, and based on the Elmore delay tested before, a more stringent timing constraint is given. Then through check experiments between conventional AAS and our constraint-oriented feedback, the effectiveness of the proposed practice on preventing over-constrain will be tested.

Table 8.3 illustrates the sink delay of routing solutions with the shortest length under different algorithms. Due to the relatively contradiction between sink delay and tree length, as well as the contrasting relationship between delays on different sinks, we can safely say that delays on some of sinks can be further reduced by increasing the

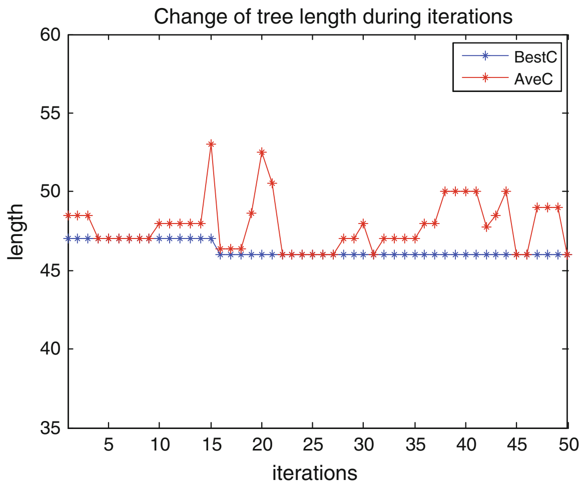


Fig. 8.14 Change of length under AAS

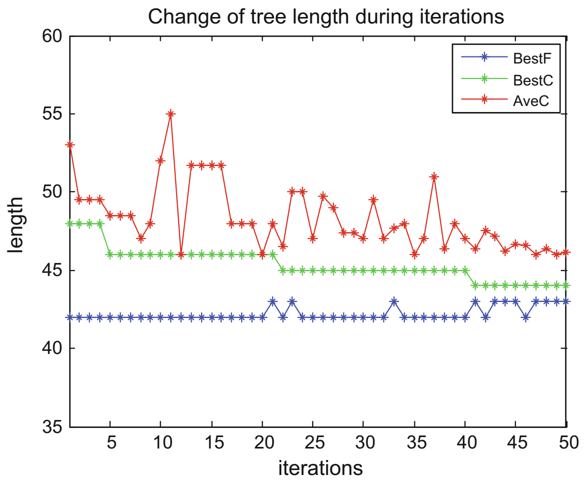


Fig. 8.15 Change of length under constraint-oriented feedback

total length or changing the topology of the final tree. Above analysis leads us to consider setting the timing constraint to be $T_{limit} = [1.5, 1.5, 1.4, 0.4, 1.5, 0.5]$.

Figures 8.14 and 8.15 depict the change of tree lengths during iterations, where *BestF* represents the length of best solutions, regardless of its violation to constraints, *BestC* represents length of best solutions that can meet the constraints, and *AveC* represents average length of solutions that can meet the constraints. If adopting AAS, only solutions under timing constraints will be reserved, and then release pheromone on corresponding paths; this procedure often requires a longer time for curves of *AveC* and *BestC* to meet, and the resulting length is not quite good. Instead, constraint-oriented feedback can take advantage of solutions that have better target value but slightly violate the constraints, by regulating little by little,

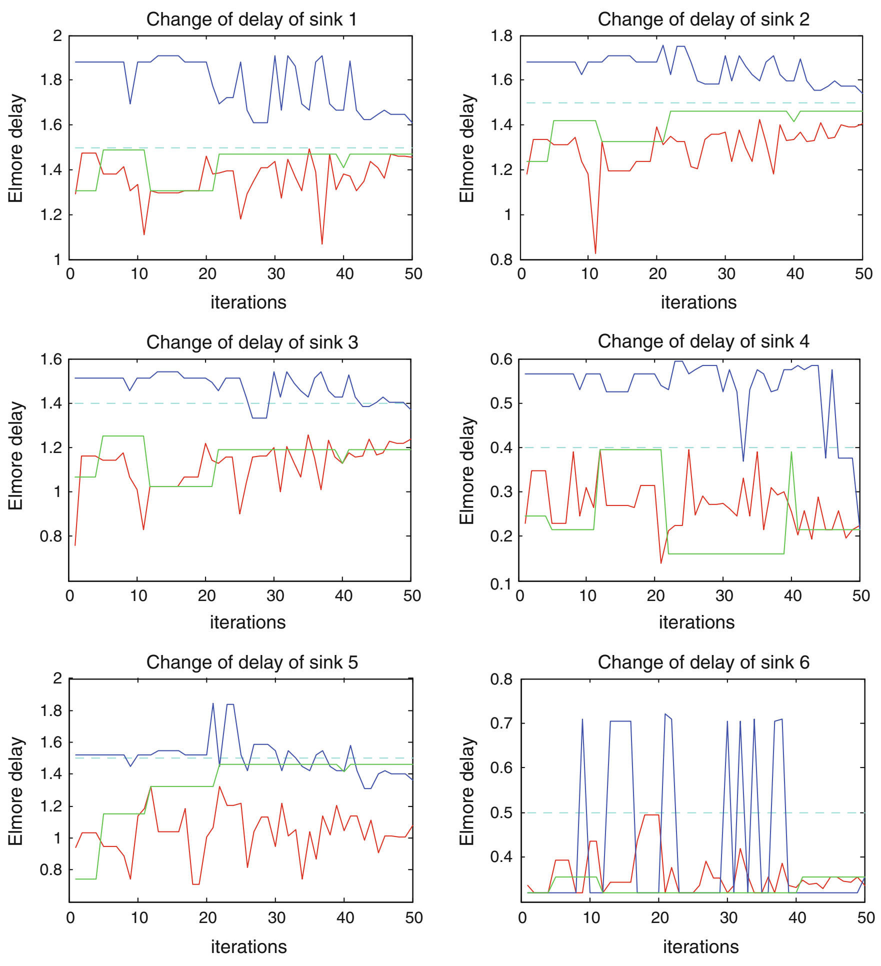


Fig. 8.16 Change of delay on each sinks under constraint-oriented feedback

also requiring quite a long process, can obtain a better solution, 44 compared to 46 in AAS. And finally, three curves in Fig. 8.15 merge together, implying that most of solutions reserved in the last iteration can satisfy the constraints so that the feedback regulation itself is converged. Figure 8.16, which depicts the change of Elmore delay of each sink during iterations, where color cyan, blue, green and, red respectively represent timing constraint, *BestF*, *BestC*, and *AveC*, also explains that points. In the early searching, the blue curves in most figures lie upon the cyan one, indicating that best solution among all feasible ones is somehow against constraints on some of its sinks. In the meanwhile, some of the green curves fall far below the cyan ones, leaving quite an allowance for improving the quality of solutions. As time goes on, the blue curve declines, so is the trend of the red one, while the green curve go through accommodations with others so as to make the overall breach

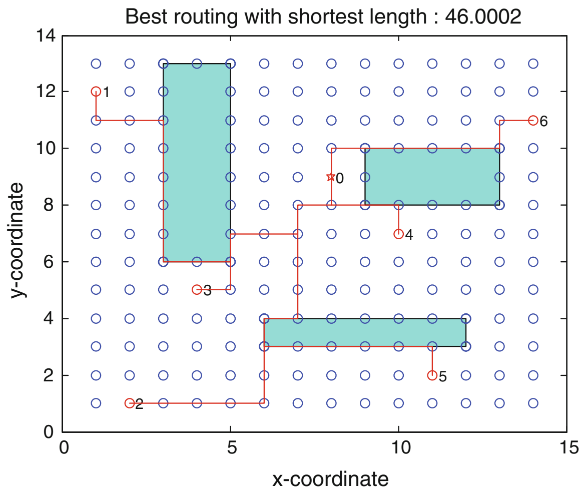


Fig. 8.17 Resulting routing diagram under AAS

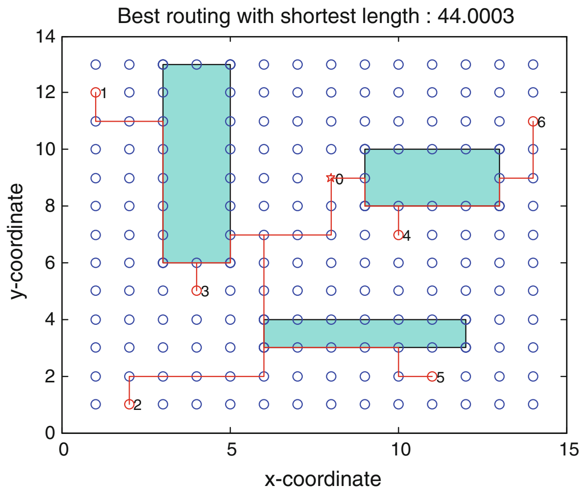


Fig. 8.18 Routing diagram under constraint-oriented feedback

smaller than before. Some sinks, like sink 4, have to larger their breaches, leaving chances for others to minish theirs. This change occurs in iterations around 5, 20, and 40, which corresponds to a step-down in *BestC* curve in Fig. 8.15.

Figures 8.17 and 8.18 give out the final routing diagram under AAS and our constraint-oriented feedback. Table 8.4 records their respective shortest lengths and their corresponding Elmore delays and breaches on each sinks. The one with the shorter length does not necessarily possess the smallest delay on every sinks, but roughly speaking, the breach of it is comparatively closer to zero. Also, shortest length is not automatically equivalent to a near-to-zero value of all elements in its vector *Breach*. Instead, an accommodation between sinks must be considered, and that's the reason why breaches under constraint-oriented feedback

Table 8.4 Comparisons between AAS and constraint-oriented feedback

Method/module	Length	Elmore delay						Breach					
		Sink 1	Sink 2	Sink 3	Sink 4	Sink 5	Sink 6	Sink 1	Sink 2	Sink 3	Sink 4	Sink 5	Sink 6
AAS	46	1.420	1.430	1.100	0.395	1.430	0.320	0.080	0.070	0.300	0.005	0.070	0.180
Feedback	44	1.470	1.460	1.190	0.215	1.460	0.355	0.030	0.040	0.210	0.185	0.040	0.045

are not always smaller than in AAS. Therefore the contradictory relationship discussed before has been once again evidenced, and the effectiveness of our constraint-oriented feedback on preventing over-constrain is convincingly demonstrated.

8.5 Summary

The global routing in VLSI belongs to the multi-terminal path planning, and can be abstracted as a MRSTRO problem. With the coming of submicron age, delay on interconnect can no longer be ignored, which makes the optimization model much different from before. Previous algorithms of constructing Steiner tree are either inapplicable or far from satisfactory. This chapter presented a novel SFB-ACO algorithm, which can serve as a useful tool for net connection with multiple endpoints under constraints. In detail, the main contributions are concluded as follows.

References

1. Chen WK (2004) The electrical engineering handbook. Academic, Burlington, MA
2. Vannelli A (1991) An adaptation of an interior point method for solving the global routing problem. IEEE Trans CAD/ICAS 10(2):193–203
3. Roy JA, Markov IL (2007) High-performance routing at the nanometer scale. In: Proceedings of the International Conference on Computer-Aided Design (ICCAD), San Jose, CA pp 496–502
4. Terlaky T, Vannelli A, Zhang H (2008) On routing in VLSI design and communication networks. Discrete Appl Math 156(11):2178–2194
5. Meindl J (2004) Tyranny of interconnects. In: Proceedings of the International Symposium on Physical Design, pp 18–21
6. Hu J, Sapatnekar S (2002) A survey on multi-net global routing for integrated circuits. Integr VLSI J 31(1):1–49
7. Courant R, Robbins H (1941) What is mathematics? Oxford University Press, New York
8. Hwang FK, Richards DS, Winter P (1992) The Steiner tree problem. Elsevier, Amsterdam
9. Winter P (1985) An algorithm for the Steiner problem in the Euclidean plane. Networks 15:323–345
10. Weng JF, Brazil M, Thomas DA, Zachariasen M (2002) Canonical forms and algorithms for Steiner trees in uniform orientation metrics. Algorithmica, Technical Report: pp 2–22
11. Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of computer computations. Plenum, New York
12. Wame DM, Winter P, Zachariasen M (1999) Exact solutions to the large scale plane Steiner tree problems. In: Proceedings of the 10th annual ACM-SIAM Symposium on Discrete Algorithms, pp 979–980
13. Snyder TL (1992) On the exact location of Steiner points in general dimension. SIAM J Comput 21(1):163–180
14. Kahng AB, Liu B (2003) Q-tree: a new iterative improvement approach for buffered interconnect optimization. In Proceedings of the IEEE Computer Society Annual Symposium on VLSI, pp 183–188

15. Boese KD, Kahng AB, McCoy BA, Robins G (1995) Near optimal critical sink routing tree constructions. IEEE Trans Comput Aided Des Integr Circ Syst 14(12):1417–1436
16. Hanan M (1966) On Steiner's problem with rectilinear distance. SIAM J Appl Math 14:255–265
17. Kastner R, Bozogzadeh E, Sarrafzadeh M (2000) Predictable routing. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, pp 110–113
18. Moffett BC (1970) Personal communication. University of Washington
19. Kastner R, Bozorgzadeh E, Sarrafzadeh M (2002) Pattern routing: use and theory for increasing predictability and avoiding coupling. IEEE Trans Comput Aided Des Integr Circ Syst 21(7):777–790
20. Westra J, Bartels C, Groeneveld P (2004) Probabilistic congestion prediction. In: Proceedings of the ACM International Symposium on Physical Design, pp 204–209
21. Hwang FK (1976) On Steiner minimal trees with rectilinear distance. SIAM J Appl Math 30:104–114
22. Nielsen BK, Winter P, Zachariasen M (2002) An exact algorithm for the uniformly-oriented Steiner tree problem. In: Proceedings of the 10th European Symposium on Algorithms. Lecture Notes in Computer Science, vol 2461. Springer, Berlin, pp 760–772
23. Borah M, Owens RM, Irwin MJ (1994) An edge based heuristic for Steiner routing. IEEE Trans Comput Aided Des Integr Circ Syst 13(12):1563–1568
24. Kahng A, Robins G (1992) A new class of iterative Steiner tree heuristics with good performance. IEEE Trans Comput-Aided Des 11:893–902
25. Kahng A, Robins G (1995) On optimal interconnects for VLSI. Kluwer Academic, Boston, MA
26. Griffith J, Robins G, Salowe JS, Zhang T (1994) Closing the gap: near-optimal Steiner trees in polynomial time. IEEE Trans Comput-Aided Des 13(11):1351–1365
27. Chu C (2004) FLUTE: fast lookup table based wire length estimation technique. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design: 696–701
28. C Chu, Y C Wong (2005) Fast and accurate rectilinear Steiner minimal tree algorithm for VLSI design. In Proceedings of the 2005 ACM International Symposium on Physical Design, pp 28–35
29. Cho M, Pan DZ (2007) BoxRouter: a new global router based on box expansion and progressive ILP. IEEE Trans Comput-Aided Des Integr Circ Syst 26:2130–2134
30. Pan M, Chu C (2006) FastRoute: a step to integrate global routing into placement. In: Proceedings IEEE/ACM International Conference on Computer-Aided Design, pp 464–471
31. Pan m, Chu C (2007) FastRoute 2.0: a high-quality and efficient global router. In: Proceedings of the 2007 Asia and South Pacific Design Automation Conference, IEEE Computer Society, pp 250–255
32. Lin CW, Chen SY, Li CF, Chang YW, Yang CL (2008) Obstacle-avoiding rectilinear Steiner tree construction based on spanning graphs. IEEE Trans Comput Aided Des Integr Circ Syst 27(4):643–653
33. Feng Z, Hu Y, Jing T, Hong X, Hu X, Yan G (2006) An O(nlogn) algorithm for obstacle-avoiding routing tree construction in the lambda-geometry plane. In: Proceedings of the 46th ACM Annual Design Automation Conference, pp 48–55
34. Shi Y, Mesa P, Yu H, He L (2006) Circuit simulation based obstacle-aware Steiner routing. Proceedings of the 43rd annual conference on Design automation, San Francisco, CA, pp 385–388
35. Shen ZC, Chu CCN, Li YM (2005) Efficient rectilinear Steiner tree construction with rectilinear blockages. In: Proceedings of the 2005 IEEE International Conference on Computer Design: VLSI in Computers and Processors, pp 38–44
36. Hare RM, Julstrom BA (2003) A spanning-tree-based genetic algorithm for some instances of the rectilinear Steiner problem with obstacles. In: Proceedings of the 2003 ACM symposium on Applied Computing, pp 725–729

37. Consoli S, Moreno-Pérez JA, Darby-Dowman K (2010) Discrete particle swarm optimization for the minimum labelling Steiner tree problem. Nat Comput 9(1): 29–46
38. Joobbani R (1985) An artificial intelligence approach to VLSI routing. PhD thesis, Carnegie-Mellon University
39. Lee JW, Choi BS, Lee JJ (2011) Energy-efficient coverage of wireless sensor networks using ant colony optimization with three types of pheromones. IEEE Trans Industr Inform 7(3):419–427
40. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization: artificial ants as computational intelligence technique. IEEE Comput Intell Mag 1(4):28–39
41. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B Cybern 26(1):29–41
42. Samanta T, Ghosal P, Dasgupta P, Rahaman H (2008) Revisiting fidelity: a case of Elmore-based y-routing tree. In: Proceedings of the 2008 ACM International Workshop on System Level Interconnect Prediction, pp 27–34
43. Dasgupta P (2005) Revisiting VLSI interconnects in deep sub-micron: some open questions. Proceedings of the IEEE 18th International Conference on VLSI Design, pp 81–86
44. McCoy BA, Boese KD, Kahng AB, Robins Gabriel (1995) Near-optimal critical sink routing tree constructions. IEEE Trans Comput-Aided Des Integr Circ Syst 14(12):1417–1436
45. Rubinstein J, Penfield P, Horowitz MA (1983) Signal delay in RC tree networks. IEEE Trans Comput-Aided Des 2:202–211
46. Hou H, Hu J, Sapatnekar SS (1999) Non-Hanan routing. IEEE Trans Comput-Aided Des Integr Circ Syst 18(4):436–444
47. McCoy BA, Boese KD, Kahng AB, Robins G (1993) Fidelity and near-optimality of Elmore-based routing constructions. Proceedings of the IEEE International Conference on Computer Design, pp 81–84
48. Prim RC (1957) Shortest connection networks and some generations. Bell Syst Tech J 36:1389–1401