Japan J. Indust. Appl. Math. (2012) 29:317–330
DOI 10.1007/s13160-012-0071-3

ORIGINAL PAPER **Area 3**

Partitioning technique for transforming perfect binary trees into single-row networks

Ser Lee Loh · Shaharuddin Salleh · Nor Haniza Sarmin

Received: 25 May 2011 / Revised: 22 February 2012 / Published online: 21 April 2012

Abstract Many problems in science and engineering can be simplified into the form of a perfect binary tree. This paper discusses our study entitled Perfect Binary Tree Sequence (PBTS) which transforms a perfect binary tree into the single-row network. The transformation is necessary in applications such as in the assignment of telephone channels to caller–receiver pairs roaming in cells in a cellular network on real-time basis. In this application, each caller and receiver in a call forms a node, while their pair connection forms the edge. A specific case of the graph in the form of a binary tree is then transformed into its corresponding single-row network for assigning the channels to the caller–receiver pairs. PBTS starts with the formation of the spine from a perfect binary tree through the insertion mechanism, and this leads to the expansion of the spine into one or more zones in the single-row network. This is followed by the formation of terminals and intervals for optimal transformation into the nets of the single-row network using our earlier method called ESSR. The numerical experiment results support our hypothesis that PBTS transforms the tree into its single-row network efficiently.

Keywords Perfect binary tree · Single-row routing · Transformation and spine

Mathematics Subject Classification 90C27

S. L. Loh · S. Salleh (✉) · N. H. Sarmin
Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia (UTM), 81310 Johor Bahru, Johor, Malaysia
e-mail: ss@utm.my

S. L. Loh
e-mail: lohserlee@yahoo.com

N. H. Sarmin
Ibnu Sina Institute For Fundamental Science Studies, UTM, 81310 Johor Bahru, Johor, Malaysia
e-mail: nhs@utm.my

1 Introduction

Single-row routing problem originates from the layout design of printed circuit boards (PCBs). The technique is one of the methods deployed for routing pins, vias and back-planes with the main objective of minimizing the congestions in the PCBs. Efficient routing ensures smooth communication between the components in the PCBs. Thus, many researches had been carried out in minimizing the congestion on the circuit board.

Single-row routing problem has been identified as an NP-complete problem [1]. Many heuristical solutions have been proposed to solve the problem as exact mathematical techniques alone are not sufficient to produce optimum solutions. For example, Ting et al. [1] and Kuh et al. [2] proposed the necessary and sufficient conditions for the optimal solutions to the single-row routing problem. In [3], a partitioning strategy was proposed to group the nets into zones which produces some reasonably good solutions for some restricted models. This work is complemented in [4] through a graph theoretical approach which relates the intervals of the single-row network with the overlap and interval graphs.

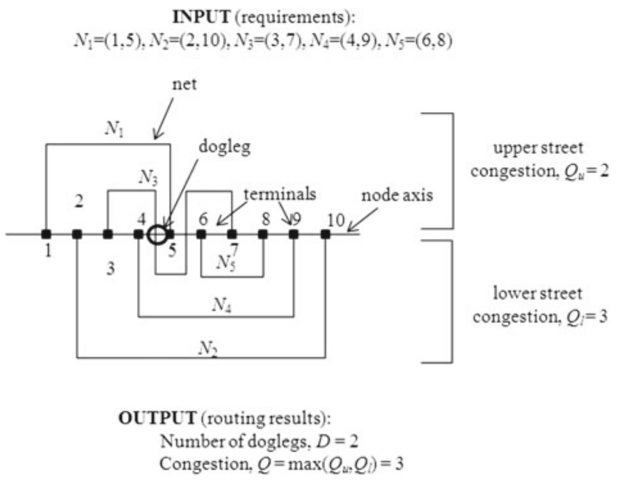
In [5], a model called *Enhanced Simulated Annealing Technique for Single-Row Routing* (ESSR) was proposed to optimize the network by minimizing both the congestion and number of doglegs. ESSR is based on the simulated annealing technique [6] which energy function is a function of the height of the segments of the nets in the single-row network. The technique has been successfully applied to produce optimal solutions to all net sizes.

Knowing that many engineering and science problems can be represented as problems in the graph theory, the relationship between a complete graph and its single-row representation was first formulated in [7,8]. Both models discuss the transformation of a graph into a single-row network where ESSR [5] is applied to produce optimal results. The transformation finds its application, for example, in assigning telephone channels to caller–receiver pairs roaming in cellular regions in a cellular network on real-time basis. In this application, each caller and receiver from a call forms a node, while their pair connection forms the edge. A specific case of the graph in the form of binary tree is then transformed into its corresponding single-row network for assigning the channels to the caller–receiver pairs.

A graph relationship with a single-row network is demonstrated through its transformation in [7]. The transformation involves the mapping of each node in a connected graph G as a zone in a single-row network S, which creates an interval for every link in the node. The set of intervals formed from the mapping then produce non-crossing nets in the single-row network S.

We propose a model called *Perfect Binary Tree Sequence* (PBTS) which performs single-row transformation through two steps: first the formation of perfect binary tree spine using the insertion mechanism, and second the expansion of the spine. PBTS is followed by ESSR for producing an optimal single-row network.

The transformation of perfect binary trees into single-row networks has a number of useful applications involving the matching problem between pairs of nodes. One such application is the processor arrangement in a parallel computing network where the processors are arranged in a single row as their axis. Another potential application



INPUT (requirements):
N1=(1,5), N2=(2,10), N3=(3,7), N4=(4,9), N5=(6,8)
net
dogleg
terminals
node axis
N1
N2
N3
N4
N5
1
2
3
4
5
6
7
8
9
10
upper street congestion, Qu=2
lower street congestion, Ql=3
OUTPUT (routing results):
Number of doglegs, D = 2
Congestion, Q = max(Qu,Ql) = 3

Fig. 1 Terminologies in the single-row routing problem

is in the assignment of radio channels for the caller–receiver pairs on real-time basis in a cellular telephone network.

This paper is organized into six sections. Section 1 is the introduction while Sect. 2 discusses the relationship between binary tree and single-row network. Section 3 is the problem statement. Our model in single-row transformation of perfect binary tree is presented in Sect. 4, while Sect. 5 presents the numerical experiment results of PBTS. Section 6 is the summary and conclusion.

2 Binary tree and single-row network

The idea for the transformation from a binary tree to the single-row network stems from our earlier work in [7]. The work involves the transformation of an arbitrary connected graph G into the single-row network S where each edge in G forms a net in S. An edge between two nodes in G is a matching between the two nodes which may represent, for example, a real-time telephone line between two users from two remote locations. Hence, the transformation from G to S has the potential of producing an optimal path between the two users for fast and uninterrupted communication.

2.1 Single-row network

Single-row routing is a combinatorial optimization problem that has been proven to be NP-complete [1,2]. Traditionally, single-row routing is one of the techniques employed for designing the routes between the electronic components of a printed-circuit board. Each path joining the terminals is called a *net*. In the single-row routing problem, we are given a set of $2m$ evenly-spaced terminals (pins or vias), t_i, for $i = 1, 2, \ldots, 2m$, arranged horizontally from left to right in a single horizontal row called the *single-row axis*. The problem is to construct m nets from the list $L = \{N_k\}$, for $k = 1, 2, \ldots, m$, formed from horizontal intervals, (b_k, e_k), in the node axis, where b_k and e_k are the beginning and end terminals of the intervals, respectively. Figure 1 shows a single-row network. Single-row network is a planar graph with each horizontal interval is formed from a pair of two terminals through non-intersecting vertical and horizontal lines. The nets are to be drawn from left to right, and a reverse direction is not allowed.

Realization is the drawing of non-crossing paths each joining a pair of terminals which aligned on a node axis. Figure 1 shows a realization from the ordering list $L = \{N_1, N_3, N_5, N_4, N_2\}$. Physically, each net in the single row represents a conductor path for its terminals to communicate. The area above the single-row axis is called the *upper street*, while the below is the *lower street*. The number of horizontal tracks in the upper and lower streets is called the *upper street congestion* Q_u and the *lower street congestion* Q_l, respectively. The overall street congestion Q of a realization is defined as the maximum of its upper and lower street congestions, that is, $Q = \max\{Q_u, Q_l\} = 3$ in the above figure. A crossing on the node axis, as shown in Fig. 1 through a line between nodes 4 and 5 in the figure, is called a *dogleg* or interstreet crossing. The realization also produces two doglegs in this example.

ESSR is based on the simulated annealing which minimizes the congestion and number of doglegs in a single-row network by minimizing the following energy function [5] which expressed as the collective properties of congestion and number of doglegs:

$$E = \sum_{i=1}^{m} \sum_{j=1}^{r_i} |h_{i,j}| \tag{1}$$

In the above equation, $|h_{i,j}|$ is the energy of segment j of net i, while r_i is number of segments in net N_i, $i = 1, 2, \ldots, m$. The lower energy value presents the better result with lower congestion and number of doglegs.

2.2 Binary tree

Binary tree is a type of tree where the degree of each vertex in the tree is at most three. A *rooted binary tree* is a tree which node has at most two children. The *root* of a tree is the node with no parent while the *leaf* node is the node which has no children. There is exactly one root node in a rooted tree. The *depth* of a node is the length of the path from the root node to the node. Hence, the root node is at depth zero. The set of nodes at a given depth is called a *level*. It follows that a *perfect binary tree* is a rooted binary tree whose leaves are located in the same depth or level. The *height*, h, of the tree is the maximum level of any of its nodes. Thus, the number of existing nodes in a perfect binary tree at level k is 2^k. A perfect binary tree with height h has $2^{h+1} - 1$ nodes and 2^h leaves. It can be implemented as an array where a node at index i has children at indexes $2i$ and $2i + 1$ and a parent at index $\frac{i}{2}$. In this paper, τ is used to represent perfect binary tree with height h.

3 Problem statement

The transformation problem can be stated as follows:

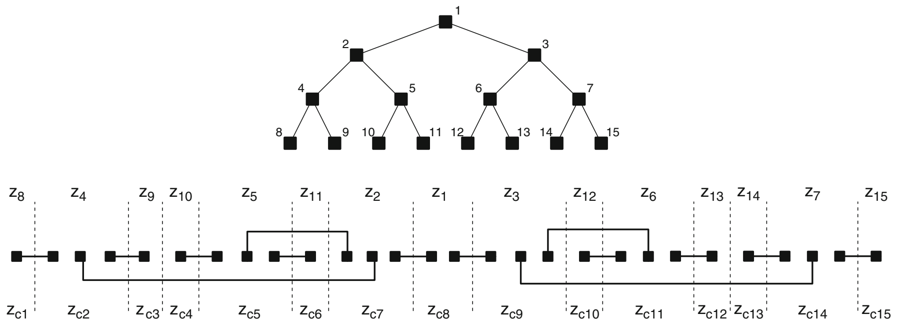

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
z_8 z_4 z_9 z_{10} z_5 z_{11} z_2 z_1 z_3 z_{12} z_6 z_{13} z_{14} z_7 z_{15}
z_{c1} z_{c2} z_{c3} z_{c4} z_{c5} z_{c6} z_{c7} z_{c8} z_{c9} z_{c10} z_{c11} z_{c12} z_{c13} z_{c14} z_{c15}

Fig. 2 Perfect binary tree τ (*up*) and its mapping into the single-row network S (*down*)

Given a connected graph in the form of a perfect binary tree τ, how can it be optimally transformed into a single-row network S so as to minimize the congestion in the network?

Figure 2 shows a perfect binary tree τ with total number of nodes, $n = 15$, and its single-row network S. A zone in S is an image of the object called node in τ while a terminal in S is a pin in the zone which works in a pair joining a net, where the net is an image of an edge in τ. The transformation from τ to S results in every node v_i in τ forming a zone z_i in S where the number of *terminals* in the zone equals to the degree d_i of the node v_i for $i = 1, 2, \ldots, 15$. For example, v_2 has degree of three and hence z_2 has three terminals. In S, the terminals are aligned evenly in a node axis where the sequence of zones in S matches node labeling in τ. The edges in τ are preserved as the nets or intervals in S. For example, v_2 is adjacent to v_4 and hence there is a net joining the terminals each one from z_2 and z_4. All nets are drawn from left to right in one way to form non-crossing paths.

Optimal transformation denotes the best transformation which gives the single-row network with the lowest congestion and number of doglegs. In order to achieve optimal transformation, arrangement of zone z_i for $i = 1, 2, \ldots, n$ is to be determined. The zones are arranged in such a way the set of shorter intervals is formed. A labeling scheme is used to assign labels $z_{c_j},\ j = 1, 2, \ldots, n$ which describes the position of zones in S. The label consists of an array, c_j, records the zone at the position j in S. For example, from Fig. 2, the first zone z_{c_1} is occupied by z_8 while the second zone z_{c_2} is occupied by z_4.

4 PBTS: the transformation model

In PBTS, a perfect binary tree is divided (starting from the leaves) into a few partitions where each partition has two levels of nodes. The *upper level nodes* and *lower level nodes* refer to the upper and lower levels of nodes respectively in the partition. A *spine* in our model refers to the partial zones in the single-row network. The partial zones correspond to the upper level nodes in each of the partition in τ. The spine is then expanded into a list of zones (except the zone which corresponds to the root node in

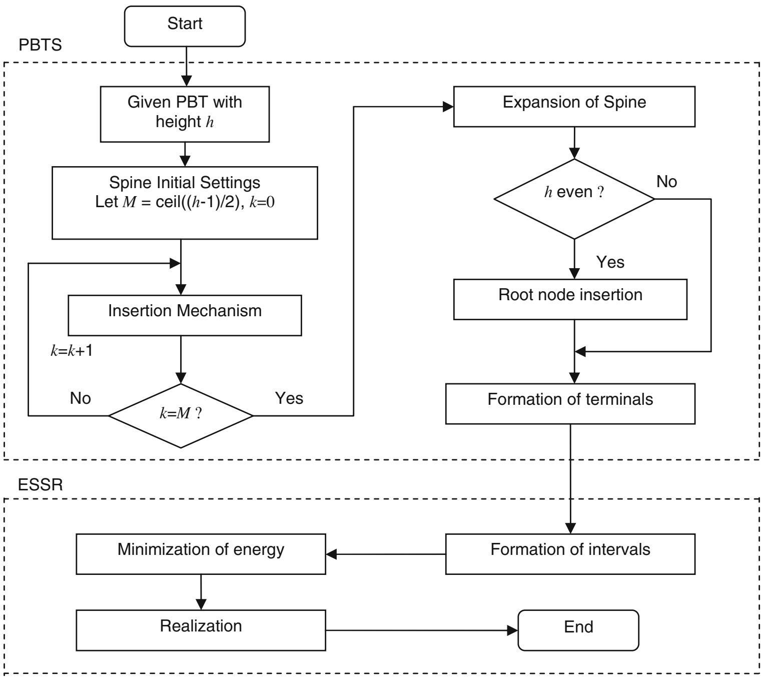


Fig. 3 The schematic flow of PBTS

perfect binary tree with even value of height) by adding all the corresponding lower level nodes in each partition.

Figure 3 shows the schematic flow of PBTS. PBTS comprises two main steps, namely, the formation of a spine from the nodes of the graph, and the expansion of the spine into single-row terminals. The transformation of a perfect binary tree τ of height h into a single-row network involves the formation of zones, terminals and intervals. In the formation of zones, we propose a new technique called Formation of Spine through Insertion Mechanism, followed by Expansion of Spine into a sequence of zones.

4.1 Formation of zones and terminals

In forming the zones, n nodes in τ are mapped into n zones in S where $n = 2^{h+1} - 1$. Each node v_i in τ has degree d_i for $i = 1, 2, \ldots, n$. Also, each zone z_i in S corresponds to node v_i in τ, and this generates the number of terminals equal to the number of degree, d_i in τ. For a perfect binary tree, the root node has the degree of two and all the leaves have the degree of one while the rest of internal nodes have the degree of three.

Arrangement of zones is crucial in the issue of optimizing the single-row transformation. PBTS involves the formation of a perfect binary tree spine with the insertion mechanism and the expansion of spine, where these algorithms will be illustrated

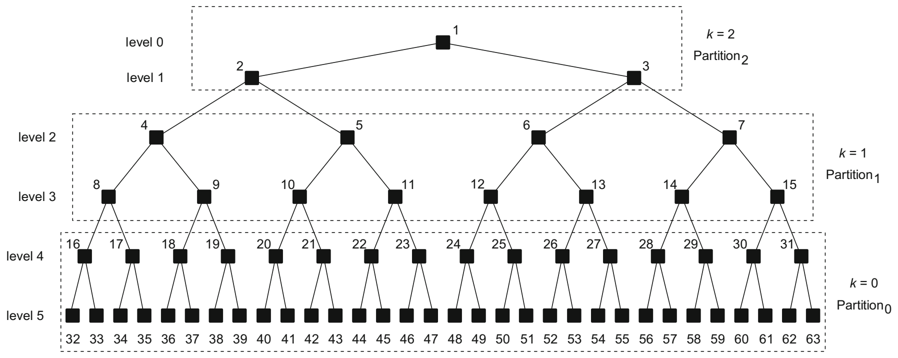


Fig. 4 A perfect binary tree τ with $h = 5$

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Fig. 5 Spine at $k = 0$

with examples. The model provides an efficient technique to compute the optimal arrangement of zones.

4.1.1 Formation of the perfect binary tree spine

In forming the spine, the perfect binary tree is divided into $M + 1$ partitions where M is the ceiling of $\frac{h-1}{2}$, or $M = \lceil \frac{h-1}{2} \rceil$. Each partition, $Partition_k$ for $k = 0, 1, \ldots, M$, is made up of two levels of nodes starting from the leaves. The first partition is $Partition_0$ which includes nodes from the bottommost two levels, while the last partition is $Partition_M$ which has the top two levels of nodes (for trees with odds h).

Figure 4 shows an example of a perfect binary tree with $h = 5$. The tree has a total of 64 nodes arranged into 6 levels with node 1 as the root, and nodes 32–63 as the leaves. There are three partitions in the tree where each partition has two levels of nodes, named upper level and lower level of nodes. All upper level of nodes in each partition are known as spine P_i.

4.1.2 Insertion mechanism

The perfect binary tree spine is formed using a technique called *Insertion Mechanism*. We refer to τ in Fig. 4 to illustrate this approach. Initially, the spine formation starts with the bottommost level nodes where $k = 0$. Each node in level 4 forms the spine, P_i for $i = 1, 2, \ldots, 2^{h-k}$as shown in Fig. 5.

The insertion mechanism starts with $k = 1$. The number of added zones which originate from the upper level nodes in partition k is determined by $A\_zone_k = 2^{h-(2k+1)}$ for $k = 1, 2, \ldots, M$. The gap between the added zones is computed as $P\_Gap_k = \frac{TtlCurrentP}{A\_zone_k}$ for $k = 1, 2, \ldots, M$ where $TtlCurrentP$ refers to the total number of zones in the spine at $(k-1)$th iteration. These zones are inserted evenly with P_Gap_k unit of gap to minimize the distance between the parents with their corresponding

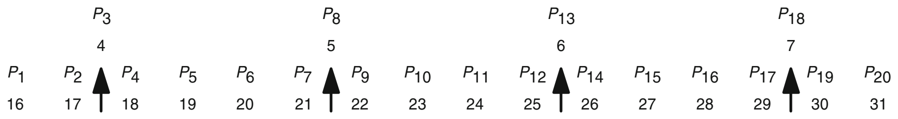


Fig. 6 Spine at $k = 1$

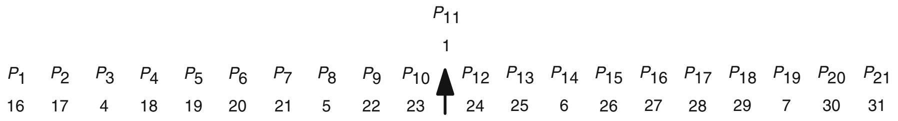


Fig. 7 Spine at $k = 2$

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
z_{cj}	z_{32}	z_{16}	z_{33}	z_{34}	z_{17}	z_{35}	z_8	z_4	z_9	z_{36}	z_{18}	z_{37}	z_{38}	z_{19}	z_{39}	z_{40}
j	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
z_{cj}	z_{20}	z_{41}	z_{42}	z_{21}	z_{43}	z_{10}	z_5	z_{11}	z_{44}	z_{22}	z_{45}	z_{46}	z_{23}	z_{47}	z_2	z_1
j	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
z_{cj}	z_3	z_{48}	z_{24}	z_{49}	z_{50}	z_{25}	z_{51}	z_{12}	z_6	z_{13}	z_{52}	z_{26}	z_{53}	z_{54}	z_{27}	z_{55}
j	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	
z_{cj}	z_{56}	z_{28}	z_{57}	z_{58}	z_{29}	z_{59}	z_{14}	z_7	z_{15}	z_{60}	z_{30}	z_{61}	z_{62}	z_{31}	z_{63}	

Fig. 8 Sequence of zones for a PBT with $h = 5$

descendants on the nodes axis in S. For example, when $k = 1,\ A\_zone_1 = 2^2 = 4$. A total of four zones are inserted evenly into the spine with $P_Gap_1$unit of distance between them, and this is shown in Fig. 6.

Since node 4 is the ancestor node of nodes 16, 17, 18 and 19, this node is inserted in the middle of its descendant nodes. This is done in order to minimize the width of intervals between node 4 and its descendant nodes in S. This strategy also contributes in minimizing the congestion and the doglegs in single-row routing problem. The next iteration follows where $k = 2,\ A\_zone_2 = 2^0 = 1$. The zone is inserted into the spine to complete the formation of spine, and this is shown in Fig. 7. The formation of PBT spine is said to be completed after M iterations.

4.1.3 Expansion of spine

All the zones in the spine are expanded to form a sequence of zones. Each node at index i has children at indexes $2i$ and $2i + 1$. From the previous example, $P_1 = 16$ has two children which are shown as nodes 32 and 33 in Fig. 8. The expansion of spine P_1 yields $z_{c_1} = z_{32},\ z_{c_2} = z_{16}$, and $z_{c_3} = z_{33}$. This process is repeated in every zone until the construction of zones is completed. The complete sequence of zones which is expanded from the spine for the previous example is shown in the Fig. 8.

For a perfect binary tree with an odd value height, the formation of zones is completed after the expansion of the spine. However, for a tree with an even height value, the formation of zones is completed by inserting the root node in the middle of the sequence of zones after the expansion of the spine.

Zero-energy net contributes towards a low congestion in the network as it can be drawn directly on the node axis. Thus, the spine is expanded in such a way to maximize the number of zero-energy nets. The formation of zones z_{c_j}, $j = 1, 2, \ldots n$ for PBT with height of h is given in Algorithm `Formation_of_Zones`.

```
Algorithm Formation_of_Zones
begin
    for $i = 1$ to $2^{h-1}$
        Compute the $i^{th}$ zone in the initial spine, $P_i = 2^{h-1} + (i-1)$;
    endfor;
    if $h \geq 3$
        for $m = 1$ to $\left\lfloor \frac{h-1}{2} \right\rfloor$
            Set $A\_zone_k = 2^{h-(2k+1)}$;
            Set the total number of zones in spine at $(k-1)^{th}$ iteration
            as $TtlCurrentP$;
            Compute $P\_Gap_k = \frac{TtlCurrentP}{A\_zone_k}$ and insert the zones evenly
            into the spine with $P\_Gap_k$ unit of gap;
        endfor;
    endif;
    Set $a = 0$
    for $k = 1$ to $2^{h-1}$
        $c_{a++} = 2P_k$;
        $c_{a++} = P_k$;
        $c_{a++} = 2P_k + 1$;
    endfor;
    if $h$ is an even number
        Insert the zone corresponds to root node in the middle of
        spine;
    endif;
end;
```

4.1.4 Formation of terminals

Algorithm `Formation_of_Terminals` outlines the formation of zones and terminals in our model. Every z_i in S corresponds to v_i in τ for $i = 1, 2, \ldots, n$. Hence, every z_i has a number of terminals equal to the degree d_i of v_i. As a result, the total number of terminals formed in S is $\sum_{i=1}^{n} d_i$. All terminals are aligned on the node axis and numbered successively following the sequence of zones, c_j for $j = 1, 2, \ldots, n$. In fact, c_j reflects the nodes label in τ.

```
Algorithm Formation_of_Terminals
begin
    Given $c_j$ as the list of zones sequence;
    /* $z_{c_j}$ in $S$ corresponds to $v_{c_j}$ in $\tau$ for $j = 1, 2, ..., n$ */
    Determine the degree, $d_{c_j}$ of every vertex, $v_{c_j}$ in $\tau$, for $j = 1, 2, ..., n$;
    Set $i = 0$;
    for $j = 1$ to $n$
        for $k = 1$ to $d_{c_j}$
            Update $i \leftarrow i + 1$;
            Set the terminal number, $t_{c_j k} = i$, in $S$;
        endfor;
    endfor;
end;
```

4.2 Formation of intervals

After the terminals are formed, the nets which connect their own pairs of terminals are constructed under the formation of intervals. All zones have their own unique corresponding nodes in τ. Zone z_i is said to be connected with z_j in S if and only if v_i is connected with v_j in τ.

There are two parts in the construction of nets. In the first part, a net is formed on a node axis between two consecutive terminals from two consecutive connected zones. This part provides the nets with zero energy, which contributes towards minimum congestion in the network.

For the second part, the list $g = \{g_1, \ldots, g_n\}$ is defined as the zone ordering for interval formation. The middle zone z_{c_j} in S is set as the first element $g_1 = j$ in the g where the nets start to form. The two zones which are next to the middle zone on the left- and the right-hand side are set to be the second and third elements arbitrarily in the g list. It follows that the two zones which are second next from the middle zone on the left- and the right-hand side are set as the fourth and fifth elements, arbitrarily. The process is repeated until the whole ordering list is completed. Figure 9 shows an example of a PBT with $h = 2$ and its g list produced using this technique.

In zone $z_{c_{g_1}}$, a net is formed between the first available terminal in it and the last available terminal in the zone on the left-hand side which is nearest and connected with $z_{c_{g_1}}$. Each zone on the left-hand side of $z_{c_{g_1}}$ is checked individually from $z_{c_{g_1}}$ in a leftward direction. The process is repeated until each of the zones on the left-hand

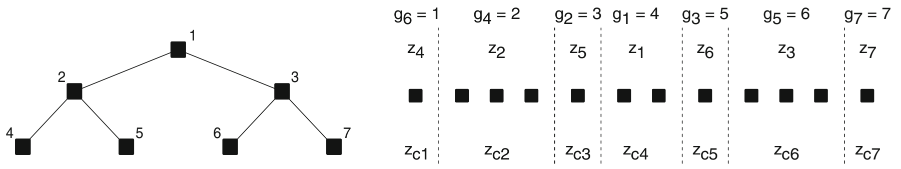


Fig. 9 PBT with $h = 2$ and its g list in S (with $n = 7$)

side which are connected to zc_{g_1} produces a net formed from zc_{g_1}. It follows then that a net will be formed between the last available terminal in zc_{g_1} and the first available terminal in the zone on the right-hand side which is nearest and connected to zc_{g_1}. Then, each zone on the right-hand side of zc_{g_1} will be checked individually from zc_{g_1} in a rightward direction. The process is repeated until each of the zones on the right-hand side which are connected to zc_{g_1} produces a net formed from zc_{g_1}.

When the net construction is completed for zc_{g_1}, the iteration is repeated for zc_{g_2} individually for n iterations until the last zone, zc_{gn} to produce a complete initial single-row network. The net construction procedure is outlined in Algorithm `Formation_of_Intervals.`

```
Algorithm Formation_of_Intervals
begin
    for $i = 1$ to $n-1$
        if zone $z_{c_i}$ is connected with zone $z_{c_{i+1}}$
            A net is formed between 2 consecutive terminals from $z_{c_i}$ and
            $z_{c_{i+1}}$;
        endif;
    endfor;
    Define a $g = \{g_1,...,g_n\}$ list as the order for interval formation of
    zones;
    Set the middle zone in $S$ as the first element in $g$;
    Complete the $g$ list with the zones which start nearest to the
    middle zone from left- and right-hand side alternately;
    for $k = 1$ to $n$
        Set $i = g_k$;
        for $j = g_k - 2$ to 1
            if $z_{c_i}$ is connected with $z_{c_j}$
                A net is formed between the last available terminal in
                $z_{c_j}$ and the first available terminal in $z_{c_i}$;
            endif;
        endfor;
        for $j = g_k + 2$ to $n$
            if $z_{c_i}$ is connected with $z_{c_j}$
                A net is formed between the last available terminal in
                $z_{c_i}$ and the first available terminal in $z_{c_j}$;
            endif;
        endfor;
    endfor;
end;
```

Once the construction process is completed, the nets are ready for sorting. The next step is renumbering the nets from their beginning terminals in ascending order. This step is followed by the assignment of each net to a unique level which represents the

Table 1 Numerical experiment results for some perfect binary trees

Height,	# nodes,	PBTS			DSA		
h	n	E	Q	D	E	Q	D
2	7	2	1	0	2	1	0
3	15	4	1	0	4	1	0
4	31	12	2	0	14	2	0
5	63	24	2	0	46	4	5
6	127	55	3	1	166	9	25
7	255	114	4	1	4, 609	26	767

ordering of nets. The final step is applying ESSR [5] to obtain the optimal sequence for the ordering of nets to produce a single-row network with the least congestion.

5 Numerical experiment results

We applied PBTS on several models with height values of between two and seven of the perfect binary tree. We also applied our earlier model, namely Double Simulated Annealing (DSA), in the same problems for comparison purpose. The sequence of zones in DSA is obtained by permutated the zones using simulated annealing without considering the properties of graphs. The results in terms of energy values (E), congestion (Q) and number of doglegs (D) are summarized in Table 1.

It is clear from the results that the formation of the perfect binary tree spine in PBTS minimizes the width of the intervals. It follows that the expansion of the spine maximizes the formation of nets with zero energy value which contributes to the low energy value in the single-row network. Figure 10a and b shows the nets ordering of τ and its final realization for both PBTS and DSA.

Regression testing is carried out for the result formulation of PBTS based on numerical experiment as shown in Table 1. The relation between the energy (E) and the order of perfect binary tree (n) from Table 1 is shown as follows:

$$E = 0.4553 - 2.6243 \tag{2}$$

The results from the simulation work are derived from the unique characteristics of a perfect binary tree, that is, all the leaves in the tree have the same depth. In the tree, the number of existing nodes at level d is 2^d. The tree with height h has $2^{h+1} - 1$ nodes and 2^h leaves. It can be efficiently presented as an array where a node at index i has children at indexes $2i$ and $2i + 1$ and a parent at index $\frac{i}{2}$. PBTS has been developed based on these unique characteristics of the tree which efficiently optimize the single-row transformation of the perfect binary tree.

6 Summary and conclusion

One difficulty in transforming a graph into a single-row network is dealing with the way the nodes from the graph are mapped as terminals in the network. The order or

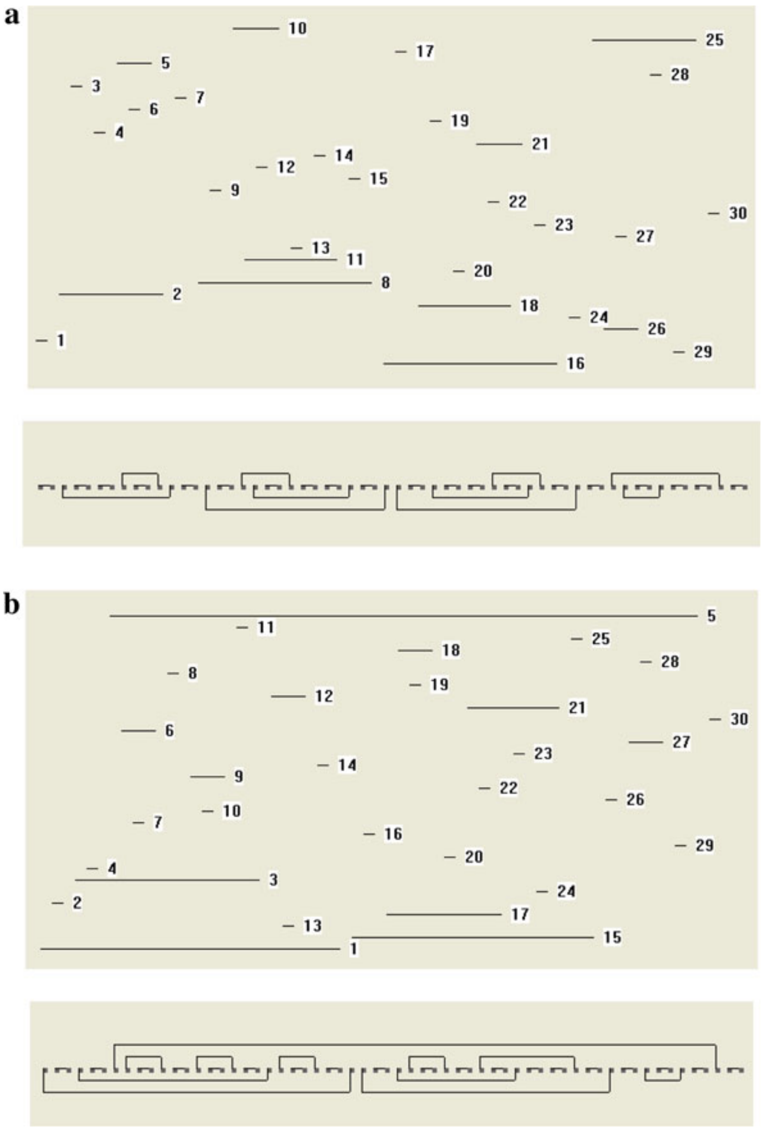


Fig. 10 **a** Nets ordering of τ (*up*) with minimum energy, $E = 12$ and its final realization (*down*) using PBTS. **b** Nets ordering of τ (*up*) with minimum energy, $E = 14$ and its final realization (*down*) using DSA

arrangement of the terminals is important as it will determine the optimality of the single-row network. Improper ordering of the terminals will definitely contribute towards high congestion in the network. Therefore, it is necessary to start the optimality strategy by dealing with the properties of the graph before transforming the nodes. We address this difficulty in this paper using the perfect binary tree as a case study.

Our work proposes a new technique for transforming a perfect binary tree into a single-row network. We propose a new model called the PBTS which includes two steps, namely, the formation of a spine using Algorithms `Formation_of_Zones and Formation_of_Terminals`, and its expansion into the terminals of the single-row network using Algorithm `Formation_of_Intervals`. The terminals produced from PBTS are arranged optimally based on the characteristics of the binary tree, and this leads to the formation of intervals or nets in the single-row network. We then apply our earlier simulation model to the intervals called ESSR to produce an optimal single-row network by minimizing the energy.

PBTS has been tested using six different models of a perfect binary tree with heights ranging from two to seven. The simulations produced optimal results in each case for minimum congestion in the final realizations.

The perfect binary tree is our first step in generalizing the technique to an arbitrary graph. It will not be easy to deal with a graph in general as different graphs have inherent properties that make them resistant to mapping. Therefore, we suggest a case by case analysis to help in achieving this objective.

Acknowledgments The authors would like to thank the Malaysian Ministry of Higher Education and Universiti Teknologi Malaysia for their financial funding through FRGS grant 78330.

References

1. Ting, B.S., Kuh, E.S., Shirakawa, L.: The multilayer routing problem: algorithms and necessary and sufficient conditions for the single row, single layer case. IEEE Trans. Circuit Syst. **23**, 768–778 (1976)
2. Kuh, E.S., Kashiwabara, T., Fujisawa, T.: On optimum single-row routing. IEEE Trans. Circuit Syst. **26**(6), 361–368 (1979)
3. Tarng, T.T., Sadowska, M.M., Kuh, E.S.: An efficient single-row algorithm. IEEE Trans. Comput. Aided Des. **3**(3), 178–183 (1984)
4. Bhattacharya, B.B., Deogun, J.S., Sherwani, N.A.: A graph theoretic approach to single row routing problems. Proc. IEEE Int. Symp. Circuit Syst. **7–9**(2), 1437–1440 (1988)
5. Salleh, S., Sanugi, B., Jamaludin, H., Olariu, S., Zomaya, A.Y.: Enhanced simulated annealing technique for the single-row routing problem. J. Supercomput. **21**(3), 285–302 (2002)
6. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–678 (1983)
7. Salleh, S., Olariu, S., Sanugi, B.: Single-row transformation of complete graphs. J. Supercomput. **31**, 265–279 (2005)
8. Salleh, S., Olariu, S., Zomaya, A.Y., Kiew, L.Y., Aziz, N.A.B.: Single-row mapping and transformation of connected graphs. J. Supercomput. **39**, 73–89 (2007)