Embedding Complete Multipartite Graphs into Certain Trees

A. Arul Shantrinal [a]   R. Sundara Rajan [a,*]

A. Ramesh Babu[a]   S. Anil[b]   Mohammed Ali Ahmed[c]

[a] Department of Mathematics,
Hindustan Institute of Technology and Science,
Chennai, India, 603 103
shandrinashan@gmail.com   vprsundar@gmail.com
arbbabu67@gmail.com

[b] Department of Computer Science and Engineering,
Hindustan Institute of Technology and Science,
Chennai, India, 603 103
samanthapudianil@gmail.com

[c] Department of Mathematics, College of Education for Pure Sciences,
University of Baghdad, Baghdad, Iraq
mohammedali1975@yahoo.com

Abstract

One of the important features of an interconnection network is its ability to efficiently simulate programs or parallel algorithms written for other architectures. Such a simulation problem can be mathematically formulated as a graph embedding problem. In this paper, we embed complete multipartite graphs into certain trees, such as k-rooted complete binary trees and k-rooted sibling trees.

Keywords: Embedding, wirelength, complete multipartite graphs, binary tree, sibling tree

1   Introduction

It is well known that a topological structure of an interconnection network can be modeled by a connected graph G. There are a lot of mutually

conflicting requirements in designing the topology of an interconnection network. It is almost impossible to design a network which is optimum from all aspects. One of the central issues in designing and evaluating an interconnection network is to study how well other existing networks can be embedded into this network and vice-versa. This problem can be mathematically formulated as a graph embedding problem [1].

A graph embedding has two main applications: to transplant parallel algorithms developed for one network to a different one, and to allocate concurrent processes to processors in the network. The quality of an embedding can be measured by certain cost criteria. One of these criteria is the *wirelength*. The wirelength of a graph embedding arises from VLSI designs, data structures and data representations, networks for parallel computer systems, biological models that deal with cloning and visual stimuli, parallel architecture, structural engineering and so on [2, 3].

Given two graphs G (guest) and H (host), an embedding from G to H is an injective mapping $f : V(G) \to V(H)$ and associating a path $P_f(e)$ in H for each edge e of G. The wirelength of an embedding $WL(G, H)$ [4] is defined as follows:

$$WL(G, H) = \min_{f:G \to H} \sum_{e=xy \in E(G)} d_H(f(x), f(y)) = \min_{f:G \to H} \sum_{e=xy \in E(H)} EC_f(e)$$

where $d_H(f(x), f(y))$ is a distance (need not be a shortest distance) between $f(x)$ and $f(y)$ in H and $EC_f(e)$ denote the number of edges e' of G such that $e = xy$ is in the path $P_f(e')$ (need not be a shortest path) between $f(x)$ and $f(y)$ in H. Further, $EC_f(S) = \sum_{e \in S} EC_f(e)$, where $S \subseteq E(H)$.

The *wirelength problem* [4, 5, 6] of a graph G into H is to find an embedding of G into H that induces the minimum wirelength $WL(G, H)$. The following version of the edge isoperimetric problem of a graph $G(V, E)$ have been considered in the literature [7], and are NP-complete [8].

For a subgraph H of G of order n,

$$E_G(H) = \{uv \in E(G) \mid u, v \in H\}, \quad E_G(k) = \max_{H \subseteq V(G),\, |H|=k} |E_G(H)|$$

The *maximum subgraph problem* (MSP) for a given k, $k \in [n]$ is a problem of computing a subset H of $V(G)$ such that $|H| = k$ and $|E_G(H)| = E_G(k)$. Further, the subsets H are called the *optimal set* [7, 8, 9].

The following results are powerful tools to find wirelength of an embedding using maximum subgraph problem.

Lemma 1.1. [10] *Let $f : G \to H$ be an embedding with $|V(G)| = |V(H)|$. Let S be set of all edges (or edge cut) of H such that $E(H) \backslash S$ has exactly two components H_1 and H_2 and let $G_i = [f^{-1}(V(H_i))]$, $i = 1, 2$. In other words, G_i is the induced subgraph on $f^{-1}(V(H_i))$ vertices, $i = 1, 2$. Moreover, S must fulfil the following conditions:*

1. *For each edge $uv \in E(G_i), i = 1, 2, P_f(uv)$ has no edges in the set S.*

2. *For each edge $uv \in E(G)$ with u in $V(G_1)$ and v in $V(G_2)$, $P_f(uv)$ has only one edge in the set S.*

3. *$V(G_1)$ and $V(G_2)$ are optimal sets.*

Then $EC_f(S)$ is minimum over all embeddings $f : G \to H$ and $EC_f(S) = \sum\limits_{v \in V(G_1)} deg_G(v) - 2|E(G_1)| = \sum\limits_{v \in V(G_2)} deg_G(v) - 2|E(G_2)|$, where $deg_G(v)$ is the degree of a vertex v in G.

Remark 1.2. For a regular graph G, it is easy to check that, $V(G_2)$ is optimal if $V(G_1)$ is optimal and vice-versa [6].

Lemma 1.3. [10] *For an embedding f from G into H, let $\{S_1, S_2, \ldots, S_p\}$ be an edge partition of $[k(H)]$ such that each S_i is an edge cut of H and it satisfies all the conditions of Lemma 1.1. Then*

$$WL_f(G, H) = \frac{1}{k} \sum_{i=1}^{p} EC_f(S_i).$$

2 Main Results

The multipartite graph is one in all the foremost in style convertible and economical topological structures of interconnection networks. The multipartite has several wonderful options and its one in all the most effective topological structure of parallel processing and computing systems. In parallel computing, a large process is often decomposed into a collection of little sub processes which will execute in parallel with communications among these sub processes. Due to these communication relations among these sub processes the multipartite graph can be applied for avoiding conflicts in the network as well as multipartite networks helps to identify the errors occurring areas in easy way. A complete p-partite graph $G = K_{n_1, \ldots, n_p}$ is a graph that contains p independent sets containing n_i, $i \in [p]$, vertices, and all possible edges between vertices from different parts.

A tree is a connected graph that contains no cycles. Trees are the most fundamental graph-theoretic models used in many fields: information theory, data structure and analysis, artificial intelligence, design of algorithms, operations research, combinatorial optimization, theory of electrical networks, and design of networks [11].

The most common type of tree is the binary tree. A binary tree is said to be a complete binary tree if each internal node has exactly two descendents. These descendents are described as left and right children of the parent node. Binary trees are widely used in data structures because they are easily stored, easily manipulated, and easily retrieved. Also, many operations such as searching and storing can be easily performed on tree data structures. Furthermore, binary trees appear in communication pattern of divide-and-conquer type algorithms, functional and logic programming, and graph algorithms. A rooted tree represents a data structure with a hierarchical relationship among its various elements [2].

There are several useful ways in which we can systematically order all nodes of a tree. Three most important ordering are called *preorder*, *inorder* and *postorder*. To achieve these orderings the tree is traversed in a particular fashion. Starting from the root, the tree is traversed counter clockwise. For preorder, we list a node the first time we pass it. For inorder, we list a leaf the first time we pass it, but list an interior node the second time we pass it. For postorder, we list a node the last time we pass it [12].

We now compute the exact wirelength of embedding complete 2^p-partite graphs $K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}}$ into k-rooted complete binary trees and k-rooted sibling trees for minimizing the wirelength, where $p, n \geq 2$. To prove the main results, we need the following result and the algorithm.

Lemma 2.1. [13] If G is a complete p-partite graph $K_{r,r,\ldots,r}$ of order pr, $p, r \geq 2$, then

$$
E_G(k) = \begin{cases}
k(k-1)/2 & ; k \leq p-1 \\
q^2 p(p-1)/2 & ; l = qp, \ 1 \leq q \leq r \\
\frac{(q-1)^2 p(p-1)}{2} + j(q-1)(p-1) + \frac{j(j-1)}{2} & ; l = (q-1)p + j, \\
& \quad 1 \leq j \leq p-1, \\
& \quad 2 \leq q \leq r.
\end{cases}
$$

Guest Graph Algorithm

Input: $N = 2^n$ (Total number of elements)

$p \geq 1$, where 2^{n-p} represents the number of elements in the each partite

Output: Labeling of complete 2^p-partite graph $K_{2^{n-p},2^{n-p},\ldots,2^{n-p}}$

1. Begin the algorithm

2. The guest graph is generated by the complete 2^p-partite graph

3. The program contains a function $disp\_3nr$ which takes $2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}$ as partite elements

4. Get the values p and N, where $p \geq 1$

5. 2^{n-p} represents the number of elements in a partite

6. 2^p represents the number of partite generated

7. **Assign elements in the partite:**

8. $m = 2^p$ //Determine number of partite

9. $y = 0$

10. for $x \leftarrow 0$ to n do

11. y ++

12. $Elem\_val = y$

13. for $i \leftarrow 0$ to p do

14. for $j \leftarrow 0$ to p do

15. $Array[x]+i+j = Elem\_val$

16. $Elem\_val = Elem\_val + N$

17. Print the partite:

18. $n = 0$ //Initiating array number

19. for $x \leftarrow 0$ to N do

20. for $i \leftarrow 0$ to p do

21. for $z \leftarrow 0$ to p do

22. for $j \leftarrow 0$ to p do

23. Print (array $[n] + i + j$))

24. Print a tab space

25. n++

26. Go to new line

27. $z = x\%p$

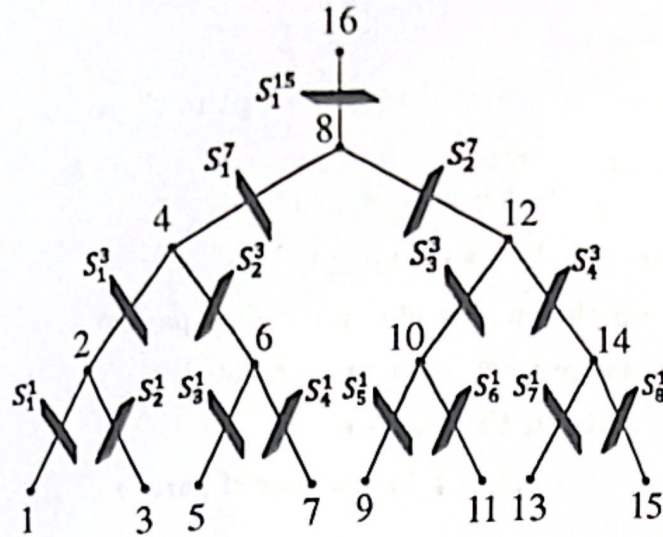28. if $z = 0$

29. Print an empty line

30. End the algorithm

Figure 1: Cut edge of 1-rooted complete binary tree T_4^1

2.1 k-rooted Complete Binary Tree

For any non-negative integer n, the complete binary tree of height n, denoted by T_n, is the binary tree where each internal vertex has exactly two children and all the leaves are at the same level. Clearly, a complete binary tree T_n has n levels and level i, $1 \leq i \leq n$, contains 2^{i-1} vertices. Thus T_n has exactly $2^n - 1$ vertices. The 1-rooted complete binary tree T_n^1 is obtained from a complete binary tree T_n by attaching to its root a pendant edge. The new vertex is called the root of T_n^1 and is considered to be at level 0. The k-rooted complete binary tree T_n^k is obtained by taking k vertex disjoint 1-rooted complete binary trees T_n^1 on 2^n vertices with roots say $n_1, n_2, \ldots, n_k$ and adding the edges (n_i, n_{i+1}), $1 \leq i \leq k-1$ [11].

Wirelength Algorithm A

Input : The complete 2^p-partite graphs $K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}}$, $p, n \geq 2$ and the 1-rooted complete binary tree T_n^1 on 2^n vertices.

Algorithm : Label the vertices of complete 2^p-partite graphs $K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}}$ using Guest Graph Algorithm. Label the vertices of T_n^1 using inorder traversal [12, 14] from 1 to 2^n, see Fig. 1. Let $f(x) = x$ for all $x \in V(K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}})$ and for $(a, b) \in E(K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}})$, let $P_f(a, b)$ be a shortest path between $f(a)$ and $f(b)$ in T_n^1.

Output : An embedding f of $K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}}$ into T_n^1 with minimum wirelength and is given by

$$WL(G, H) = \begin{cases} 3n2^n - 2^{2n+1} + 2^{n-p}(2^p - 1)(n2^n - 2^n + 1) + 2 & ; \quad j \le p \\ \\ 2^{n-p}(2^p - 1)(n2^n - 32^n + 2n + 3) & ; \quad j > p. \end{cases}$$

Proof of correctness : For $j = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, 2^{n-j}$, let $S_i^{2^j - 1}$ be the cut edge of the 1-rooted complete binary tree T_n^1, which has one vertex in level $n - j$ and the other vertex in level $n - j + 1$, such that $S_i^{2^j - 1}$ disconnects T_n^1 into two components $H_i^{2^j - 1}$ and $\overline{H}_i^{2^j - 1}$ where $V(H_i^{2^j - 1}) = \{2^j(i-1) + 1, 2^j(i-1) + 2, \ldots, 2^j(i-1) + 2^j - 1\}$, see Fig. 1. Let $G_i^{2^j - 1}$ and $\overline{G}_i^{2^j - 1}$ be the inverse images of $H_i^{2^j - 1}$ and $\overline{H}_i^{2^j - 1}$ under f respectively. By the Guest Graph Algorithm and Lemma 2.1, $G_i^{2^j - 1}$ is an optimal set in $K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}}$. Thus the cut edge $S_i^{2^j - 1}$ satisfies all the conditions of Lemma 1.1. Therefore $EC_f(S_i^{2^j - 1})$ is minimum for $j = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, 2^{n-j}$ and is given by

$$EC_f(S_i^{2^j - 1}) = \begin{cases} (2^j - 1)(2^{n-p}(2^p - 1) - (2^j - 2)) & ; \quad j \le p \\ \\ (2^p - 1)(2^{n-p}(2^j - 1) - 2^{j-p}(2^j - 2)) & ; \quad j > p. \end{cases}$$

Then by Lemma 1.3,

$$\begin{aligned} WL(G, H) &= \sum_{j=1}^{n} \sum_{i=1}^{2^{n-j}} EC_f(S_i^{2^j - 1}) \\ \\ &= \begin{cases} \sum_{j=1}^{n} \sum_{i=1}^{2^{n-j}} (2^j - 1)(2^{n-p}(2^p - 1) - (2^j - 2)) & ; \quad j \le p \\ \\ \sum_{j=1}^{n} \sum_{i=1}^{2^{n-j}} (2^p - 1)(2^{n-p}(2^j - 1) - 2^{j-p}(2^j - 2)) & ; \quad j > p \end{cases} \\ \\ &= \begin{cases} 3n2^n - 2^{2n+1} + 2^{n-p}(2^p - 1)(n2^n - 2^n + 1) + 2 & ; \quad j \le p \\ \\ 2^{n-p}(2^p - 1)(n2^n - 32^n + 2n + 3) & ; \quad j > p. \end{cases} \end{aligned}$$

Wirelength Algorithm B

Input : The complete 2^p-partite graphs $K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}}$, $p, n \ge 2$ and the k-rooted complete binary tree $T_{n_1}^k$, $k = 2^{n-n_1}$.

Algorithm : Label the vertices of complete 2^p-partite graphs $K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}}$ using Guest Graph Algorithm. Label the vertices of $T_{n_1}^k$, $k = 2^{n-n_1}$ as follows:
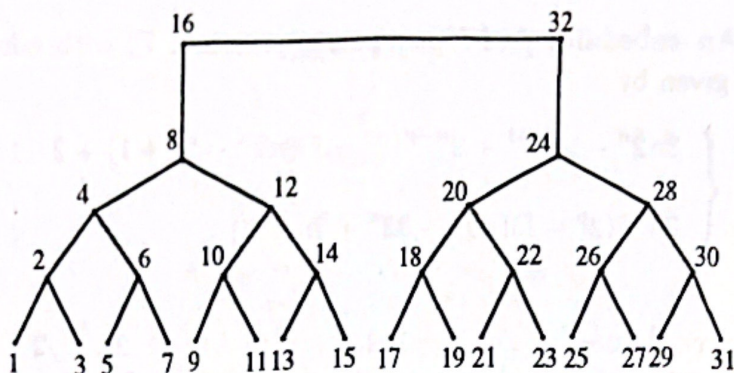
Figure 2: Inorder labeling of 2-rooted complete binary tree T_4^2

Let $T_{n_1}^{1,1}, T_{n_1}^{1,2}, \ldots, T_{n_1}^{1,k}$ be the k vertex disjoint 1-rooted complete binary trees of T_n^k. Label the vertices of $T_{n_1}^{1,i}$, $1 \leq i \leq k$ using inorder traversal [12, 14], see Fig. 2. Let $f(x) = x$ for all $x \in V(K_{2^{n-p},2^{n-p},\ldots,2^{n-p}})$ and for $(a,b) \in E(K_{2^{n-p},2^{n-p},\ldots,2^{n-p}})$, let $P_f(a,b)$ be a shortest path between $f(a)$ and $f(b)$ in $T_{n_1}^k$.

Output : An embedding f of $K_{2^{n-p},2^{n-p},\ldots,2^{n-p}}$ into $T_{n_1}^k$ with minimum wirelength and is given by

$$
WL(G,H) = k \cdot \begin{cases} 3n_1 2^{n_1} - 2^{2n_1+1} \\ \quad + 2^{n-p}(2^p-1)(n_1 2^{n_1} - 2^{n_1} + 1) + 2 & ; j \leq p \\[2ex] 2^{n_1-p}(2^p-1) \\ \quad (2n_1 - 2^{n_1+1} + n_1 2^n - 2^n + 2^{n-n_1} + 2) & ; j > p \end{cases}
$$

$$
+ \begin{cases} \frac{1}{6}[k(k-1)2^{n_1-p} \\ \quad (3\,2^p - 3\,2^n + 2^{n_1+p} + 3\,2^{n+p} - k\,2^{n_1+p+1})] & ; i2^{n_1} \leq 2^p \\[2ex] \frac{1}{6}[k(k-1)2^{n_1-p}(2^p-1) \\ \quad (2^{n_1} - k\,2^{n_1+1} + 3\,2^n)] & ; i2^{n_1} > 2^p. \end{cases}
$$

Proof of correctness : By Wirelength Algorithm A, it is enough to prove that the cut edge (n_i, n_{i+1}), $1 \leq i \leq k-1$, where n_i is the root of $T_{n_1}^{1,i}$, $1 \leq i \leq k$, has minimum edge congestion. The cut edge (n_i, n_{i+1}), $1 \leq i \leq k-1$ of $T_{n_1}^k$, disconnects $T_{n_1}^k$ into two components H_i and $\overline{H}_i$ where $V(H_i) = \{1, 2, \ldots, i2^{n_1}\}$. Let G_i and $\overline{G}_i$ be the inverse images of H_i and $\overline{H}_i$ under f respectively. By the Guest Graph Algorithm and by Lemma 2.1, G_i is an optimal set in $K_{2^{n-p},2^{n-p},\ldots,2^{n-p}}$. Thus the cut edge (n_i, n_{i+1}), $1 \leq i \leq k-1$ satisfies all the conditions of Lemma 1.1. Therefore $EC_f((n_i, n_{i+1}))$ is minimum for $i = 1, 2, \ldots, k-1$ and is given by

$$
EC_f((n_i, n_{i+1})) = \begin{cases} i2^{n_1}(2^{n-p}(2^p-1) - i2^{n_1} + 1) & ; & i2^{n_1} \leq 2^p \\[2ex] i2^{n_1-p}(2^p-1)(2^n - i2^{n_1}) & ; & i2^{n_1} > 2^p. \end{cases}
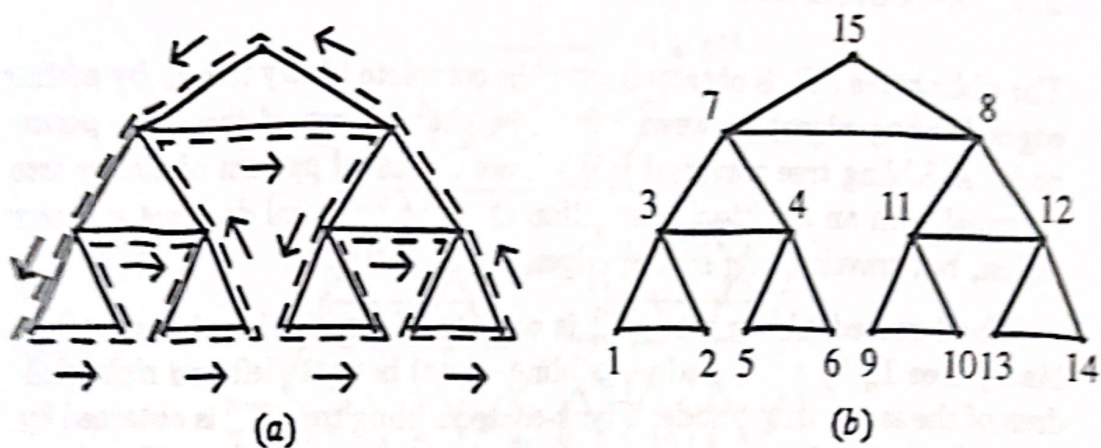$$

Figure 3: (a) Sibling tree traversal (b) Labeling of ST_5 using sibling tree traversal

Then by Lemma 1.3,

$$WL(G,H) = k \cdot \sum_{j=1}^{n_1} \sum_{i=1}^{2^{n_1-j}} EC_f(S_i^{2^j-1}) + \sum_{i=1}^{k-1} EC_f(n_i, n_{i+1})$$

$$= k \cdot \begin{cases} \sum_{j=1}^{n_1} \sum_{i=1}^{2^{n_1-j}} (2^j-1)(2^{n-p}(2^p-1)-(2^j-2)) & ; \quad j \le p \\[2em] \sum_{j=1}^{n_1} \sum_{i=1}^{2^{n_1-j}} (2^p-1)(2^{n-p}(2^j-1)-2^{j-p}(2^j-2)) & ; \quad j > p \end{cases}$$

$$+ \begin{cases} \sum_{i=1}^{k-1} i2^{n_1}(2^{n-p}(2^p-1)-i2^{n_1}+1) & ; \quad i2^{n_1} \le 2^p \\[1.5em] \sum_{i=1}^{k-1} i2^{n_1-p}(2^p-1)(2^n-i2^{n_1}) & ; \quad i2^{n_1} > 2^p \end{cases}$$

$$= k \cdot \begin{cases} 3n_1 2^{n_1} - 2^{2n_1+1} \\ +2^{n-p}(2^p-1)(n_1 2^{n_1} - 2^{n_1}+1)+2 & ; \quad j \le p \\[1.5em] 2^{n_1-p}(2^p-1) \\ (2n_1 - 2^{n_1+1} + n_1 2^n - 2^n + 2^{n-n_1}+2) & ; \quad j > p \end{cases}$$

$$+ \begin{cases} \frac{1}{6}[k(k-1)2^{n_1-p} \\ (3 \cdot 2^p - 3 \cdot 2^n + 2^{n_1+p} + 3 \cdot 2^{n+p} - k \cdot 2^{n_1+p+1})] & ; \quad i2^{n_1} \le 2^p \\[1.5em] \frac{1}{6}[k(k-1)2^{n_1-p}(2^p-1) \\ (2^{n_1} - k \cdot 2^{n_1+1} + 3 \cdot 2^n)] & ; \quad i2^{n_1} > 2^p. \end{cases}$$

281

2.2 k-rooted Sibling Tree

The sibling tree ST_n is obtained from the complete binary tree T_n by adding edges (sibling edges) between left and right children of the same parent node. A Sibling tree traversal [15] follows the usual pattern of binary tree traversal with an additional condition that the traversal does not cut any region, but travels along sibling edges, see Fig. 3(a).

The 1-rooted sibling tree ST_n^1 is obtained from the 1-rooted complete binary tree T_n^1 by adding edges (sibling edges) between left and right children of the same parent node. The k-rooted sibling tree ST_n^k is obtained by taking k copies of vertex disjoint 1-rooted sibling tree ST_n^1 on 2^n vertices with roots say $n_1, n_2, \ldots, n_k$ and adding the edges (n_i, n_{i+1}), $1 \le i \le k-1$ [15].

Wirelength Algorithm C

Input : The complete 2^p-partite graphs $K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}}$, $p, n \ge 2$ and the 1-rooted sibling tree ST_n^1.

Algorithm : Label the vertices of $K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}}$ using Guest Graph Algorithm. Label the vertices of ST_n^1 using sibling tree traversal from 1 to 2^n, see Fig. 3(b). Let $f(x) = x$ for all $x \in V(K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}})$ and for $(a, b) \in E(K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}})$, let $P_f(a, b)$ be a shortest path between $f(a)$ and $f(b)$ in ST_n^1.

Output : An embedding f of $K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}}$ into ST_n^1 with minimum wirelength and is given by

$$
WL(G, H) = \begin{cases} 3n2^{n-1} - 2^{2n} + 2^{n-p-1} \\ (2^p - 1)(n2^n - 2^n + 1) + 1 & ; \quad j \le p \\[2ex] 2^{n-p-1}(2^p - 1)(n2^n - 32^n + 2n + 3) & ; \quad j > p \end{cases}
$$

$$
+ \begin{cases} 2^{n-1}(5n - 2^{n+1} - 4) \\ -2^{n-p-1}(2^p - 1)(n2^n - 2) + 3 & ; \quad j+1 \le p \\[2ex] 2^{n-p-1}(2^p - 1) \\ (n2^n - 4\,2^n + 4n + 2) - n + 1 & ; \quad j+1 > p. \end{cases}
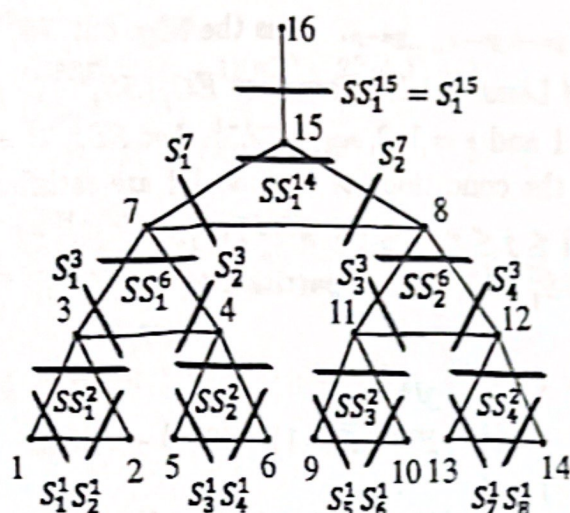$$

$$
+ 2^{n-p-1}(2^p - 1).
$$

282

Figure 4: Edge cut of 1-rooted sibling tree ST_4^1

Proof of correctness : For $j = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, 2^{n-j}$, let $S_i^{2^j-1}$ be an edge cut of the 1-rooted sibling tree ST_n^1 consisting of edges induced by the $\lceil i/2 \rceil^{\text{th}}$ parent vertex from left to right in level $n-j$ with its left child if i is odd and its right child if i is even together with the corresponding sibling edge which is the same edge in either case, such that $S_i^{2^j-1}$ disconnects ST_n^1 into two components $H_i^{2^j-1}$ and $\overline{H_i}^{2^j-1}$ where $V(H_i^{2^j-1})$ is consecutively labeled [14], see Fig. 4. Let $G_i^{2^j-1}$ and $\overline{G_i}^{2^j-1}$ be the inverse images of $H_i^{2^j-1}$ and $\overline{H_i}^{2^j-1}$ under f respectively. By Guest Graph Algorithm and Lemma 2.1, $G_i^{2^j-1}$ is an optimal set in $K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}}$. Thus the edge cut $S_i^{2^j-1}$ satisfies all the conditions of Lemma 1.1. Therefore $EC_f(S_i^{2^j-1})$ is minimum for $j = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, 2^{n-j}$ and is given by

$$EC_f(S_i^{2^j-1}) = \begin{cases} (2^j - 1)(2^{n-p}(2^p - 1) - (2^j - 2)) & ; \quad j \leq p \\ (2^p - 1)(2^{n-p}(2^j - 1) - 2^{j-p}(2^j - 2)) & ; \quad j > p. \end{cases}$$

For $j = 1, 2, \ldots, n-1$ and $i = 1, 2, \ldots, 2^{n-j-1}$, let $SS_i^{2(2^j-1)}$ be an edge cut of the 1-rooted sibling tree ST_n^1 consisting of the edges induced by the ith parent vertex from left to right in level $n-j$ and its two children, such that $SS_i^{2(2^j-1)}$ disconnects ST_n^1 into two components $H_i^{2(2^j-1)}$ and $\overline{H_i}^{2(2^j-1)}$ where $V(H_i^{2(2^j-1)})$ is consecutively labeled [14], see Fig. 4. Let $G_i^{2(2^j-1)}$ and $\overline{G_i}^{2(2^j-1)}$ be the inverse images of $H_i^{2(2^j-1)}$ and $\overline{H_i}^{2(2^j-1)}$ under f respectively. By Guest Graph Algorithm and Lemma 2.1, $G_i^{2(2^j-1)}$ is an

283

optimal set in $K_{2^{n-p}, 2^{n-p}, \ldots, 2^{n-p}}$. Thus the edge cut $SS_i^{2(2^j-1)}$ satisfies all the conditions of Lemma 1.1. Therefore $EC_f(SS_i^{2(2^j-1)})$ is minimum for $j = 1, 2, \ldots, n-1$ and $i = 1, 2, \ldots, 2^{n-j-1}$. Let $SS_1^{2^n-1} = S_1^{2^n-1}$ and it is easy to see that the conditions of Lemma 1.1 are satisfied. We note that the set $\{S_i^{2^j-1} : 1 \le j \le n, 1 \le i \le 2^{n-j}\} \cup \{SS_i^{2(2^j-1)} : 1 \le j \le n-1, 1 \le i \le 2^{n-j-1}\} \cup \{SS_1^{2^n-1}\}$ forms a partition of $[2E(ST_n^1)]$. Further,

$$
EC_f(SS_i^{2(2^j-1)}) =
\begin{cases}
(2^{j+1} - 2) \\
(2^{n-p}(2^p - 1) - (2^{j+1} - 3)) & ; \quad j+1 \le p \\
\\
(2^p - 1)(2^{n-p}(2^{j+1} - 2) \\
-2^{j-p+2}(2^j - 2)) - 2 & ; \quad j+1 > p.
\end{cases}
$$

Then by Lemma 1.3,

$$
\begin{aligned}
WL(G, H) &= \frac{1}{2}\Big[\sum_{j=1}^{n}\sum_{i=1}^{2^{n-j}} EC_f(S_i^{2^j-1}) \\
&\quad + \sum_{j=1}^{n-1}\sum_{i=1}^{2^{n-j-1}} EC_f(SS_i^{2(2^j-1)}) + EC_f(SS_1^{2^n-1})\Big]
\end{aligned}
$$

$$
= \frac{1}{2}
\begin{cases}
\sum_{j=1}^{n}\sum_{i=1}^{2^{n-j}} (2^j - 1) \\
(2^{n-p}(2^p - 1) - (2^j - 2)) & ; \quad j \le p \\
\\
\sum_{j=1}^{n}\sum_{i=1}^{2^{n-j}} (2^p - 1) \\
(2^{n-p}(2^j - 1) - 2^{j-p}(2^j - 2)) & ; \quad j > p
\end{cases}
$$

$$
+ \frac{1}{2}
\begin{cases}
\sum_{j=1}^{n-1}\sum_{i=1}^{2^{n-j-1}} (2^{j+1} - 2) \\
(2^{n-p}(2^p - 1) - (2^{j+1} - 3)) & ; \quad j+1 \le p \\
\\
\sum_{j=1}^{n-1}\sum_{i=1}^{2^{n-j-1}} (2^p - 1)(2^{n-p}(2^{j+1} - 2) \\
-2^{j-p+2}(2^j - 2)) - 2 & ; \quad j+1 > p.
\end{cases}
$$

$$
+ \frac{1}{2} 2^{n-p}(2^p - 1)
$$

$$= \begin{cases} \begin{aligned} & 3n2^{n-1} - 2^{2n} \\ & +2^{n-p-1}(2^p-1)(n2^n - 2^n + 1) + 1 \end{aligned} \quad ; \quad j \le p \\ \\ 2^{n-p-1}(2^p-1)(n2^n - 32^n + 2n + 3) \quad ; \quad j > p \end{cases}$$

$$+ \begin{cases} \begin{aligned} & 2^{n-1}(5n - 2^{n+1} - 4) \\ & -2^{n-p-1}(2^p-1)(n2^n - 2) + 3 \end{aligned} \quad ; \quad j+1 \le p \\ \\ \begin{aligned} & 2^{n-p-1}(2^p-1) \\ & (n2^n - 4\,2^n + 4n + 2) - n + 1 \end{aligned} \quad ; \quad j+1 > p. \end{cases}$$

$$+ \; 2^{n-p-1}(2^p-1).$$

As $V(ST_{n_1}^k) = V(T_{n_1}^k)$, $k = 2^{n-n_1}$, using the proof techniques of Wirelength Algorithm B and Wirelength Algorithm C, we have the following result.

Theorem 2.2. *The embedding of complete 2^p-partite graphs $K_{2^{n-p},2^{n-p},...,2^{n-p}}$, $p,n \ge 2$ into k-rooted sibling tree $ST_{n_1}^k$, $k = 2^{n-n_1}$, induces a minimum wirelength $WL(K_{2^{n-p},2^{n-p},...,2^{n-p}}, ST_{n_1}^k)$.*

3 Concluding Remarks

In this paper, we have obtained the wirelength of embedding complete multipartite graphs into certain tree derived architecture, such as k-rooted complete binary tree and k-rooted sibling trees. Finding the other parameters such as dilation and congestion of embedding complete multipartite graphs into the graphs discussed in this paper are under investigation.

References

[1] J.M. Xu and M. Ma, *Survey on path and cycle embedding in some networks*, Frontiers of Mathematics in China, 4, (2009), 217–252.

[2] J.M. Xu, *Topological Structure and Analysis of Interconnection Networks*, Kluwer Academic Publishers, (2001).

[3] Y.L. Lai and K. Williams, *A survey of solved problems and applications on bandwidth, edgesum, and profile of graphs*, Journal of Graph Theory, 31, (1999), 75–94.

[4] S.L. Bezrukov, J.D. Chavez, L.H. Harper, M. Röttger and U.P. Schroeder, *Embedding of hypercubes into grids*, Mathematical Foundations of Computer Science, (1998), 693–701.

[5] S.L. Bezrukov, J.D. Chavez, L.H. Harper, M. Röttger and U.P. Schroeder, *The congestion of n-cube layout on a rectangular grid*, Discrete Mathematics, 213, (2000), 13–19.

[6] P. Manuel, I. Rajasingh, B. Rajan and H. Mercy, *Exact wirelength of hypercube on a grid*, Discrete Applied Mathematics, 157(7), (2009), 1486–1495.

[7] S.L. Bezrukov, S.K. Das and R. Elsässer, *An edge-isoperimetric problem for powers of the Petersen graph*, Annals of Combinatorics, 4, (2000), 153–169.

[8] M.R. Garey and D.S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Freeman, San Francisco (1979).

[9] L.H. Harper, *Global Methods for Combinatorial Isoperimetric Problems*, Cambridge University Press, (2004).

[10] M. Miller, R.S. Rajan, N. Parthiban and I. Rajasingh, *Minimum linear arrangement of incomplete hypercubes*, The Computer Journal, 58(2), (2015) 331–337.

[11] I. Rajasingh, P. Manuel, B. Rajan and M. Arockiaraj, *Wirelength of hypercubes into certain trees*, Discrete Applied Mathematics, 160, (2012), 2778 - 2786.

[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, *Introduction to Algorithms*, MIT Press and McGraw-Hill, New York, (2001).

[13] R.S. Rajan, T.M. Rajalaxmi, J.B. Liu and G. Sethuraman, *Wirelength of embedding complete multipartite graphs into certain graphs*, Discrete Applied Mathematics, (2018). https://doi.org/10.1016/j.dam.2018.05.034

[14] I. Rajasingh, J. Quadras, P. Manuel and A. William, *Embedding of cycles and wheels into arbitrary trees*, Networks, 44, (2004), 173–178.

[15] R.S. Rajan, I. Rajasingh, P. Manuel, M. Miller and T.M. Rajalaxmi, *Maximum incomplete recursive circulants in graph embeddings*, Discrete Mathematics, Algorithms and Applications, 7, (2015), 1–22.