Proceedings of International Conference
on Computer Architecture

A Shortest-Path Routing Algorithm for Incomplete WK-Recursive Networks

Ming-Yang Su and Gen-Huey Chen
Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, TAIWAN

Dyi-Rong Duh
Department of Electronic Engineering,
Hwa Shia Junior College, Taipei, TAIWAN

Abstract

Designing shortest-path routing algorithms for incomplete networks is in general more difficult than for complete networks. The reason is that most incomplete networks lack a unified representation. One of the contributions of this paper is to demonstrate a useful representation, i.e., the multistage graph representation, for the incomplete WK-recursive networks. On the basis of it, a shortest-path routing algorithm is then proposed. With $O(d{\cdot}t)$ time preprocessing, this algorithm takes $O(t)$ time for each intermediate node to determine the next node along the shortest path, where $d>1$ is the size of the basic building block and $t\geq 1$ is the level of expansion.

1 Introduction

One crucial step on designing a large-scale multiprocessor system is to determine the topology of the interconnection network (network for short). In the recent decade, a number of networks have been proposed in the literature [1, 3, 10, 12, 17]. Among them, the WK-recursive networks own two structural advantages: expansibility and equal degree. A network is *expansible* if no changes to node configuration and link connection are necessary when it is expanded, and of *equal degree* if its nodes have the same degree no matter what the size is. A network with these two properties will gain the advantages of easy implementation and low cost when it is manufactured.

Although the WK-recursive networks own many favorable properties (see [2, 4, 5-6, 17, 18]), there is a rigorous restriction on their sizes. As will become clear in the next section, the number of nodes contained in a WK-recursive network must satisfy d^t, where $d>1$ is the size of the basic building block and $t\geq 1$ is the level of expansion. Thus, as $d=4$, extra $3{\cdot}4^7=49152$ nodes are required to expand from a 7-level WK-recursive network to an 8-level one. Almost all announced networks have suffered from the same restriction. To relieve this restriction, some incomplete networks such as incomplete hypercubes [7], incomplete star networks [8, 11], incomplete rotator graphs [9] and incomplete WK-recursive networks [13], have been defined recently.

In [7], a shortest-path routing algorithm for the incomplete hypercubes was presented. In [8], Latifi and Bagherzadeh suggested a heuristic routing algorithm for the incomplete star networks. Their routing algorithm is not a shortest-path one. Ravikumar *et al.* [11], on the other hand, presented a shortest-path routing algorithm for a special class of the incomplete star networks that comprise c k-stars, where $1<c\leq k$. Although shortest-path routing algorithms have been proposed for the star networks [1] and the rotator graphs [3], no shortest-path routing algorithms exist for the incomplete star networks and the incomplete rotator graphs. In this paper, a shortest-path routing algorithm is designed for the incomplete WK-recursive networks. With $O(d{\cdot}t)$ time preprocessing, this algorithm takes $O(t)$ time for each intermediate node to determine the next node along the shortest path.

In the next section, we formally define the incomplete WK-recursive networks and show that their structures can be conveniently expressed as multistage graphs. Section 3 shows a prerequisite step for our routing algorithm. The stages are grouped into consecutive blocks such that each block contains one or more stages, and every two adjacent blocks share a common stage. The union of all blocks is exactly the set of the stages. In Section 4, a heuristic routing algorithm is presented. By its aid, a shortest-path routing algorithm is presented in Section 5. Finally, this paper is concluded in Section 6.

2 Incomplete WK-recursive networks and their multistage graph representations

In this section we first review the WK-recursive networks. The incomplete WK-recursive networks are defined as their induced subgraphs (graphs and networks are used interchangeably in this paper). It is shown that the incomplete WK-recursive networks can be conveniently expressed as multistage graphs. Some necessary notations and definitions are also introduced.

The WK-recursive networks can be constructed incrementally by grouping basic building blocks together. Any complete graph can serve as a basic building block. Let $K(d, t)$ denote a WK-recursive network of level t whose basic building blocks are each a d-node complete graph, where $d>1$ and $t\geq 1$. K(d, 1), which is the basic building block, is a d-node complete graph, and K(d, t) for $t\geq 2$ can be constructed by connecting d K(d, t-1)'s as a d-supernode complete graph (each K(d, t-1) is regarded as a supernode). Each node of K(d, t) is associated with a t-digit identifier. The following definition is due to Chen and Duh [2].

Definition 2.1. The node set of K(d, t) is denoted by $\{a_{t-1}a_{t-2}...a_1a_0 \mid a_i \in \{0, 1, ..., d-1\}$ for $0\leq i\leq t-1\}$. Node adjacency is defined as follows: $a_{t-1}a_{t-2}...a_1a_0$ is adjacent to (1) $a_{t-1}a_{t-2}...a_1b$, where $0\leq b\leq d-1$ and $b\neq a_0$, and (2) $a_{t-1}a_{t-2}...a_{j+1}a_{j-1}(a_j)^j$ if $a_j\neq a_{j-1}$ and $a_{j-1}=a_{j-2}= ... =a_0$ for some $1\leq j\leq t-1$, where $(a_j)^j$ represents j consecutive a_j's. The links of (1) are called *substituting links*, and are labeled 0. The link of (2), if existing, is called a *j-flipping link* (or simply *flipping link*), and is labeled j. Besides, if $a_{t-1}=a_{t-2}= ... =a_0$, there is a link, called *open link*, incident to node $a_{t-1}a_{t-2}...a_1a_0$. The open link is labeled t. Since the open link is reserved for further expansion, its other end node is unspecified.

K(d, t) contains d^t nodes. Since each node is incident with d-1 substituting links and one flipping link (or open link), K(d, t) has degree d. The structures of K(4, 1) and K(4, 3) are illustrated in Figure 1. The substituting links are those within basic building blocks, whereas the links connecting two embedded K(d, j)'s are j-flipping links. The open links are left for future expansion. For illustration let us consider the incident links of node 311 in Figure 1. The one to node 133 is a 2-flipping link; the others are substituting links.

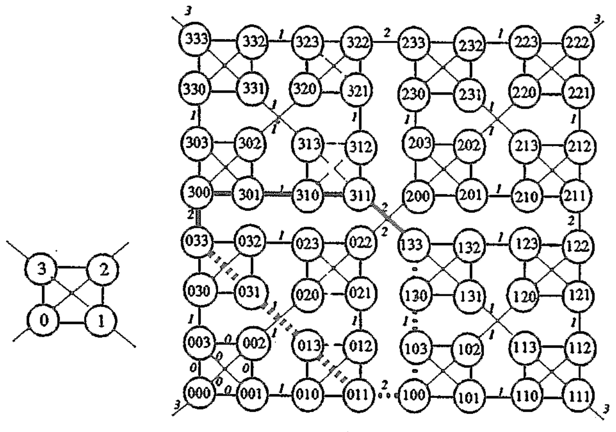
Figure 1. The structures of K(4, 1) and K(4, 3). This figure also shows two routing paths between 033 and 133.

Definition 2.2. Define $c_{t-1}c_{t-2}...c_m$·K(d, m) to be the subgraph of K(d, t) induced by $\{c_{t-1}c_{t-2}...c_ma_{m-1}...a_1a_0 \mid a_j \in \{0, 1, ..., d-1\}$ for $0\leq j\leq m-1\}$, where $1\leq m<t$ and c_{t-1}, c_{t-2}, ..., c_m are all integers from $\{0, 1, ..., d-1\}$. That is, $c_{t-1}c_{t-2}...c_m$·K(d, m) is an embedded K(d, m) with identifier $c_{t-1}c_{t-2}...c_m$.

For example, refer to Figure 1, where 31·K(4, 1) is the subgraph of K(4, 3) induced by {310, 311, 312, 313}.

Definition 2.3. Node $a_{t-1}a_{t-2}...a_1a_0$ is a *k-frontier* if $a_{k-1}= ... =a_1=a_0$, where $1\leq k\leq t$.

By definition a k-frontier is automatically an l-frontier for $1\leq l<k$. Both end nodes of a k-flipping link are k-frontiers. An embedded K(d, m) contains one (m+1)-frontier and d-1 m-frontiers. These d frontiers are 2^m-1 distant from each other.

The incomplete WK-recursive networks, which were originally defined in [13], are induced subgraphs of the WK-recursive networks. If we number the nodes of K(d, t) according to their lexicographical order, then an N-node incomplete WK-recursive network is the subgraph of K(d, t) induced by the first N nodes. Throughout this paper, we use IK(d, t) to denote an N-node incomplete WK-recursive network, where $d^{t-1}<N<d^t$ and N is a multiple of d.

The *coefficient vector* of IK(d, t) is defined as the (t-1)-vector (b_{t-1}, b_{t-2}, ..., b_1) such that $N=b_{t-1}d^{t-1}+b_{t-2}d^{t-2}+ ... +b_1d$, where $0\leq b_m\leq d-1$ for all $1\leq m\leq t-1$. IK(d, t) with coefficient vector (b_{t-1}, b_{t-2}, ..., b_1) contains b_m embedded K(d, m)'s with identifiers $b_{t-1}b_{t-2}...b_{m+1}0$, $b_{t-1}b_{t-2}...b_{m+1}1$, ..., and $b_{t-1}b_{t-2}...b_{m+1}(b_m-1)$, respectively, where $1\leq m\leq t-1$. For example, IK(5, 10) with coefficient vector (4, 2, 4, 3, 4, 1, 1, 3, 1) contains the following embedded K(d, m)'s.

0·K(5, 9), 1·K(5, 9), 2·K(5, 9), 3·K(5, 9)
40·K(5, 8), 41·K(5, 8)
420·K(5, 7), 421·K(5, 7), 422·K(5, 7), 423·K(5, 7)
4240·K(5, 6), 4241·K(5, 6), 4242·K(5, 6)
42430·K(5, 5), 42431·K(5, 5), 42432·K(5, 5), 42433·K(5, 5)
424340·K(5, 4)
4243410·K(5, 3)
42434110·K(5, 2), 42434111·K(5, 2), 42434112·K(5, 2)
424341130·K(5, 1)

Figure 2 shows the structure of IK(4, 3) with coefficient vector (3, 2). In the rest of this paper, coefficient vector (b_{t-1}, b_{t-2}, ..., b_1) is written as (b_{t-1}, b_{t-2}, ..., b_i, *), where $1\leq i\leq t-1$, provided $b_i\neq 0$ and $b_{i-1}=b_{i-2}= ... =b_1=0$. For example, (2, 0, 4, 0, 0) is written as (2, 0, 4, *), and (2, 3, 4, 1, 1) is written as (2, 3, 4, 1, 1, *).

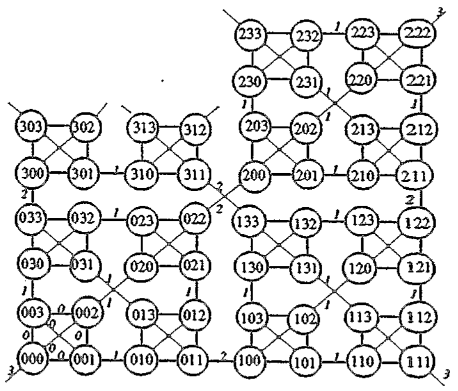
Figure 2. The structures of IK(4, 3) with coefficient vector (3, 2).

Let S_m represent the subgraph of IK(d, t) with coefficient vector (b_{t-1}, b_{t-2}, ..., b_i, $*$) induced by the nodes of $b_{t-1}b_{t-2}...b_{m+1}0\cdot$K($d$, m), $b_{t-1}b_{t-2}...b_{m+1}1\cdot$K($d$, m), ..., and $b_{t-1}b_{t-2}...b_{m+1}(b_m\text{-}1)\cdot$K($d$, m), where $i \le m \le t\text{-}1$. That is, S_m contains b_m embedded K(d, m)'s with identifiers $b_{t-1}b_{t-2}...b_{m+1}0$, $b_{t-1}b_{t-2}...b_{m+1}1$, ..., and $b_{t-1}b_{t-2}...b_{m+1}(b_m\text{-}1)$, respectively. For example, when IK(5, 10) with coefficient vector (4, 2, 4, 3, 4, 1, 1, 3, 1, $*$) is concerned, S_9 contains 0·K(5, 9), 1·K(5, 9), 2·K(5, 9), and 3·K(5, 9), S_8 contains 40·K(5, 8) and 41·K(5, 8), and S_2 contains 42434110·K(5, 2), 42434111·K(5, 2), and 42434112·K(5, 2). We note that there is an m-flipping link between any two of the embedded K(d, m)'s within S_m, where $i \le m \le t\text{-}1$. If each S_m is regarded as a stage, then the structure of the IK(d, t) forms a (t-i)-stage graph, denoted by $S_{t-1}+S_{t-2}+ ... +S_i$. For example, refer to Figure 3 where the structure of IK(5, 10) with coefficient vector (4, 2, 4, 3, 4, 1, 1, 3, 1, $*$) is represented by a 9-stage graph. For simplicity each embedded K(d, m) within S_m is drawn as a circle, and the one with identifier $b_{t-1}b_{t-2}...b_{m+1}j$, where $0 \le j \le b_m\text{-}1$, is denoted by C_m^j. All the links within S_m are omitted for conciseness.

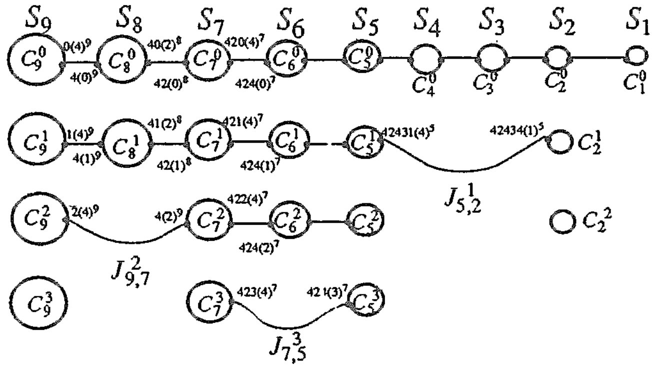

Figure 3. Multistage graph representation of IK(5, 10) with coefficient vector (4, 2, 4, 3, 4, 1, 1, 3, 1,$*$).

There are min{b_m, b_{m-1}} n-flipping links between S_m and S_{m-1} that connect C_m^j and C_{m-1}^j for $0 \le j \le \min\{b_m, b_{m-1}\}\text{-}1$. Besides, there may exist a u-flipping link between S_u and S_v, where $t\text{-}1 \ge u > v \ge i$ and $u\text{-}v>1$. Such a link, if existing, is called a *jumping u-flipping link*. As shown below, a necessary and sufficient condition for the existence of jumping flipping links has been suggested in [14].

Theorem 2.1.[14] For IK(d, t) with coefficient vector (b_{t-1}, b_{t-2}, ..., b_i, $*$), one jumping u-flipping link exists between S_u and S_v if and only if $b_u > b_{u-1} = b_{u-2} = ... = b_{v+1} < b_v$, where $t\text{-}1 \ge u > v \ge i$ and $u\text{-}v>1$. Moreover, this jumping flipping link connects C_u^e and C_v^e, where $e = b_{u-1} = b_{u-2} = ... = b_{v+1}$.

Let $J_{u,v}^e$ represent the jumping u-flipping link connecting C_u^e and C_v^e (refer to Figure 3 for illustration). Theorem 2.1 provides a simple method to determine all jumping flipping links from the coefficient vector (b_{t-1}, b_{t-2}, ..., b_i, $*$). We only need to examine (b_{t-1}, b_{t-2}, ..., b_i, $*$) from the left to the right, and $J_{u,v}^e$ exists if and only if $b_u > b_{u-1} = b_{u-2} = ... = b_{v+1} < b_v$, where $u\text{-}v>1$ and $e = b_{u-1} = b_{u-2} = ... = b_{v+1}$.

The structure of $S_{t-1}+S_{t-2}+ ... +S_i$ is further explained as follows. Since each C_m^j is a K(d, m), the links inside C_m^j are subject to Definition 2.1. On the other hand, the links incident to C_m^j include (1) b_m-1 m-flipping links connecting C_m^0, C_m^1, ..., C_m^{j-1}, C_m^{j+1}, ..., and $C_m^{b_m-1}$, respectively; (2) one m-flipping link connecting C_{m-1}^j if $j \le b_{m-1}\text{-}1$, or one jumping m-flipping link connecting C_l^j, where $l<m\text{-}1$, if $j = b_{m-1} = b_{m-2} = ... = b_{l+1} < b_l$; (3) one ($m$+1)-flipping link connecting C_{m+1}^j if $j \le b_{m+1}\text{-}1$, or one jumping h-flipping link connecting C_h^j, where $h>m+1$, if $b_h > b_{h-1} = b_{h-2} = ... = b_{m+1} = j$. Both end nodes of (1) are $b_{t-1}b_{t-2}...b_{m+1}j(x)^m \in C_m^j$ and $b_{t-1}b_{t-2}...b_{m+1}x(j)^m \in C_m^x$, where $0 \le x \le b_m\text{-}1$ and $x \ne j$. Both end nodes of (2) are $b_{t-1}b_{t-2}...b_{m+1}j(b_m)^m \in C_m^j$ and $b_{t-1}b_{t-2}...b_{m+1}b_m(j)^m \in C_{m-1}^j$ (or $\in C_l^j$). Both end nodes of (3) are $b_{t-1}b_{t-2}...b_{m+2}j(b_{m+1})^{m+1} \in C_{m+1}^j$ (or $b_{t-1}b_{t-2}...b_{h+1}j(b_h)^h \in C_h^j$) and $b_{t-1}b_{t-2}...b_{m+2}b_{m+1}(j)^{m+1} \in C_m^j$.

We note that for $i \le m \le t\text{-}1$ and $b_m \ne 0$, $S_m+S_{m-1}+ ... +S_i$ forms an embedded IK(d, m+1) with coefficient vector (b_m, b_{m-1}, ..., b_i, $*$) whose each node has its identifier prefixed with $b_{t-1}b_{t-2}...b_{m+1}$. For example, refer to Figure 3 where $S_5+S_4+S_3+S_2+S_1$ forms an embedded IK(5, 6) with coefficient vector (4, 1, 1, 3, 1, $*$) whose each node has its identifier prefixed with 4243. Theorem 2.1 can be applied to $S_m+S_{m-1}+ ... +S_i$ as well.

3 A prerequisite step

In this section, a prerequisite step for our routing algorithm is described. More concretely, an algorithm is presented to group the stages into consecutive blocks such that each block contains one or more stages, and every two adjacent blocks share a common stage. The union of all blocks is exactly the set of the stages. The algorithm will be invoked by our routing algorithm.

The input to the algorithm is a coefficient vector (b_m, b_{m-1}, ..., b_i, $*$) together with an integer λ, where $i \le m \le t\text{-}1$ and $0 \le \lambda \le b_m\text{-}1$. The output is a sequence of integers m_0, m_1, ..., m_k, where $k \ge 0$ and $m \ge m_0 > m_1 > ... > m_k = i$. The meaning of the output is that the stages S_m, S_{m-1}, ..., S_i are grouped into k+1 blocks, i.e., $S_m+S_{m-1}+ ... +S_{m_0}$, $S_{m_0}+S_{m_0-1}+ ... +S_{m_1}$, ..., and $S_{m_{k-1}}+S_{m_{k-1}-1}+ ... +S_{m_k}$. To simplify notations and without losing generality, we explain the algorithm by letting $m=t-1$. The algorithm, as shown below, takes $O(t)$ time.

Algorithm *Stage_Grouping*$((b_{t-1}, b_{t-2}, ..., b_i, *), \lambda)$. /* $0 \le \lambda \le b_{t-1}-1$ */

1. Scan $(b_{t-1}, b_{t-2}, ..., b_i, *)$ from the left to the right and determine in sequence $J^{x_1}_{y_1,z_1}, J^{x_2}_{y_2,z_2}, ..., J^{x_n}_{y_n,z_n}$ such that $\lambda > x_1 > x_2 > ... > x_n$. That is, $J^{x_1}_{y_1,z_1}$ is the first jumping flipping link encountered in the scanning which has $x_1 < \lambda$. Each $J^{e}_{u,v}$ between $J^{x_j}_{y_j,z_j}$ and $J^{x_{j+1}}_{y_{j+1},z_{j+1}}$, where $1 \le j < n$, has $e \ge x_j$, and each $J^{e}_{u,v}$ after $J^{x_n}_{y_n,z_n}$ has $e \ge x_n$. Then store $x_1, x_2, ..., x_n$ in a linked list L. If no feasible jumping flipping link is found in the scanning, L is empty.

For example, refer to Figure 3 where the coefficient vector is (4, 2, 4, 3, 4, 1, 1, 3, 1, *). If $\lambda=3$, then two feasible jumping flipping links, i.e., $J^{2}_{9,7}$ and $J^{1}_{5,2}$, are determined, and L contains two values 2 and 1. If $\lambda=2$, only $J^{1}_{5,2}$ is determined, and L contains one single value 1. If $\lambda=1$ or 0, no feasible jumping flipping link is found, and L is empty. Also note that $t-1 \ge y_1 > z_1 \ge y_2 > z_2 \ge ... \ge y_n > z_n \ge i$. By the aid of Theorem 2.1, this step can be completed in $O(t)$ time.

2. Determine $m_0=\min\{r \mid b_r > \lambda \text{ and } b_j \ge \lambda \text{ for } t-1 \ge j \ge r\}$. If L contains $x_1 > x_2 > ... > x_n$, determine $m_1, m_2, ..., m_n$ sequentially as follows: $m_1=\min\{r \mid b_r > x_1 \text{ and } b_j \ge x_1 \text{ for } m_0 > j > r\}$, $m_2=\min\{r \mid b_r > x_2 \text{ and } b_j \ge x_2 \text{ for } m_1 > j > r\}$, ..., and $m_n=\min\{r \mid b_r > x_n \text{ and } b_j \ge x_n \text{ for } m_{n-1} > j > r\}$.

By examining $(b_{t-1}, b_{t-2}, ..., b_i, *)$ from the left to the right, this step can be completed in $O(t)$ time. Refer to Figure 3 again. If $\lambda=3$, we have $m_0=9$, $m_1=5$, and $m_2=2$. If $\lambda=2$, we have $m_0=5$ and $m_1=2$. If $\lambda=1$, we have $m_0=2$. If $\lambda=0$, we have $m_0=1$. Also note that $t-1 \ge m_0 \ge y_1 > z_1 \ge m_1 \ge y_2 > z_2 \ge ... \ge m_{n-1} \ge y_n > z_n \ge m_n \ge i$.

3. Output $(m_0, m_1, ..., m_k)$ for some $k \ge 0$ according to the following four cases.
   Case 1. L is empty and $m_0=i$. Output (m_0).
   Case 2. L is empty and $m_0>i$. Set $m_1=i$ and Output (m_0, m_1).
   Case 3. L is not empty and $m_n=i$. Output $(m_0, m_1, ..., m_n)$.
   Case 4. L is not empty and $m_n>i$. Set $m_{n+1}=i$ and Output $(m_0, m_1, ..., m_n, m_{n+1})$.

Note that $k=n$ or $n+1$, and $m_k=i$. For the example of Figure 3, (9, 5, 2, 1) is output if $\lambda=3$, (5, 2, 1) is output if $\lambda=2$, (2, 1) is output if $\lambda=1$, and (1) is output if $\lambda=0$.

Clearly the time complexity of the algorithm is $O(t)$. The output $(m_0, m_1, ..., m_k)$ defines $k+1$ blocks, denoted by $B_0, B_1, ..., B_k$, where $B_0=S_{t-1}+S_{t-2}+ ... +S_{m_0}$ and $B_r=S_{m_{r-1}}+S_{m_{r-1}-1}+ ... +S_{m_r}$ for $1 \le r \le k$. Any two adjacent blocks B_{r-1} and B_r contain one common stage $S_{m_{r-1}}$. In the following lemma, we use x_0 to represent λ.

Lemma 3.1.[14] Let $J^{x_1}_{y_1,z_1}, J^{x_2}_{y_2,z_2}, ..., J^{x_n}_{y_n,z_n}$ and $m_0, m_1, ..., m_n$ be defined as in Algorithm *Stage_Grouping*. Then, for $1 \le j \le n$,
(1) $m_{j-1} \ge y_j$;
(2) if $m_{j-1} > y_j$, then $b_{m_{j-1}} > (x_{j-1} \ge) b_{m_{j-1}-1} \ge ... \ge b_{y_j}$;
(3) $z_j \ge m_j$;
(4) if $z_j > m_j$, then $b_{z_j} > x_j$, $b_{m_j} > x_j$, and $b_q \ge x_j$ for $z_j > q > m_j$;
(5) $x_j = \min\{b_{m_{j-1}}, b_{m_{j-1}-1}, ..., b_{y_j}, ..., b_{z_j}, ..., b_{m_j}\}$.

See [14] for the proof. To help understanding, this lemma is illustrated with Figure 4 where $m_{j-1} > y_j$ and $z_j > m_j$ are assumed. Two blocks B_{j-1} and B_j are shown, which intersect with $S_{m_{j-1}}$.

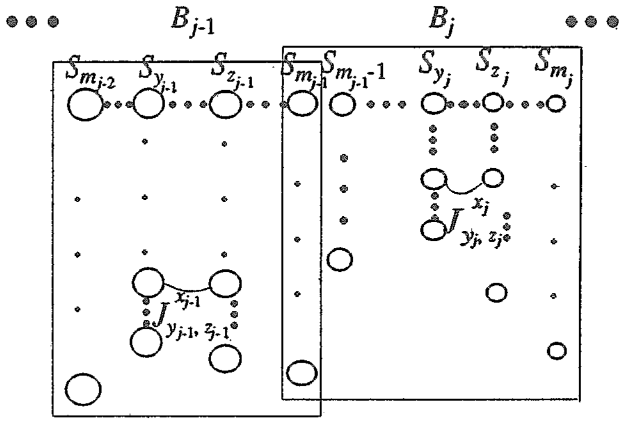

Figure 4. Illustration of Lemma 3.1.

According to Lemma 3.1, $J^{x_1}_{y_1,z_1}, J^{x_2}_{y_2,z_2}, ..., J^{x_n}_{y_n,z_n}$ are known to be the leftmost and upmost jumping flipping links in $B_1, B_2, ..., B_n$, respectively (the smaller the value x_j is, the upper $J^{x_j}_{y_j,z_j}$ is). That is, for any $J^{e}_{u,v}$ in B_j we have $u \le y_j$ and $e \ge x_j$ (in fact, we have $u \le z_j$ if $J^{e}_{u,v} \ne J^{x_j}_{y_j,z_j}$). We note that B_0 may or may not contain jumping flipping links, and B_{n+1}, if existing, does not contain any jumping flipping link. To say more precisely, for B_{n+1} we have $b_{m_n} > (x_n \ge) b_{m_n-1} \ge ... \ge b_{m_{n+1}}$ $(m_{n+1}=i)$. We also note that B_0 contains at least one stage, B_j for $0 < j \le n$ contains at least three stages, and B_{n+1}, if existing, contains at least two stages.

4 A heuristic routing algorithm

In this section, we present a heuristic routing algorithm for IK(d, t). The length of the routing path can be determined in $O(t)$ time. It is shown that the routing path is the shortest for some sepcial cases. In the rest of this paper, we use X

and Y to denote the source node and the destination node, respectively.

First of all, we have to review Vecchia and Sanges's [17] routing algorithm for K(d, t) because it will be invoked by our algorithm. For any two nodes A and B in K(d, t), we define $A =_r B$ if they belong to the same embedded K(d, r), and $A \neq_r B$ otherwise, where $1 \leq r \leq t$. For example, refer to Figure 1 where $300 =_2 321$, but $300 \neq_1 321$. A routing path from X to Y within K(d, t) can be determined by the following procedure [17].

1. Determine the level r, where $1 \leq r \leq t$, such that $X =_r Y$ and $X \neq_{r-1} Y$.
2. Determine the flipping link, say (W, Z), such that $X =_{r-1} W$ and $Z =_{r-1} Y$.
3. Determine the routing path from X to W, and the routing path from Z to Y, all in a rescursive manner.

A routing path from X to Y is obtained by concatenating the routing path from X to W, the flipping link (W, Z), and the routing path from Z to Y. For example, the routing path from node 033 to node 133 within K(4, 3) is shown with bold lines in Figure 1. It has been shown in [2] that the maximum length of the routing paths is not greater than D_t, where $D_t = 2^t - 1$ is the diameter of K(d, t). The routing algorithm, although simple, does not guarantee the shortest path. For example, the shortest path from 033 to 133 is shown with dashed lines in Figure 1. Let $p(X, Y)$ denote the routing path from X to Y within K(d, t) that is obtained by Vecchia and Sanges's algorithm. Chen and Duh [2] have proven the following three lemmas.

Lemma 4.1.[2] Suppose X and Y are two nodes of K(d, t). If $X =_r Y$ and X (or Y) is an r-frontier, where $1 \leq r \leq t$, then $p(X, Y)$ is the shortest.

Lemma 4.2.[2] Suppose $X = x_{t-1} x_{t-2} \ldots x_1 x_0$ and $Y = y_{t-1} y_{t-2} \ldots y_1 y_0$ are two nodes of K(d, t). If $X =_r Y$ and X (or Y) is an r-frontier, where $1 \leq r \leq t$, then $p(X, Y)$ has length equal to $\sum_{\substack{0 \leq j \leq t-1 \\ x_j \neq y_j}} 2^j$.

Lemma 4.3.[2] Suppose $X = x_{t-1} x_{t-2} \ldots x_1 x_0$ and $Y = y_{t-1} y_{t-2} \ldots y_1 y_0$ are two nodes of K(d, t). If $X =_r Y$ and both X and Y are r-frontiers, where $1 \leq r \leq t$, then $p(X, Y)$ has length equal to $D_r = 2^r - 1$.

By Lemmas 4.1, 4.2, and 4.3, the distance between X and Y can be determined in $O(t)$ time if $X =_r Y$ and either X or Y is an r-frontier, and $O(1)$ time if $X =_r Y$ and both X and Y are r-frontiers.

Now we are ready to describe the routing algorithm for IK(d, t). Suppose the IK(d, t) has coefficient vector (b_{t-1}, b_{t-2}, ..., b_i, $*$), and without loss of generality, assume $X \in C_f^{\alpha}$ and $Y \in C_g^{\beta}$, where $i \leq f \leq t-1$, $i \leq g \leq t-1$, $0 \leq \alpha \leq b_f - 1$, and $0 \leq \beta \leq b_g - 1$. To simplify our discussion, we further assume $f = t-1$. Let $ip(X, Y)$ denote the path that the routing algorithm traverses from X to Y. The routing algorithm is to be explained by means of the construction of $ip(X, Y)$. If $g = t-1$, we simply let $ip(X, Y) = p(X, Y)$. That is, $ip(X, Y)$ is constructed as a concatenation of $p(X, W)$, (W, Z), and $p(Z, Y)$, where $W \in C_{t-1}^{\alpha}$, $Z \in C_{t-1}^{\beta}$, and (W, Z) is the flipping link between C_{t-1}^{α} and C_{t-1}^{β}.

If $g < t-1$, with (b_{t-1}, b_{t-2}, ..., b_i, $*$) and α as input Algorithm *Stage_Grouping* is executed to produce (m_0, m_1, ..., m_k), where $k \geq 0$ and $m_k = i$. So, we have $B_0 = S_{t-1} + S_{t-2} + \ldots + S_{m_0}$, $B_1 = S_{m_0} + S_{m_0-1} + \ldots + S_{m_1}$, ..., and $B_k = S_{m_{k-1}} + S_{m_{k-1}-1} + \ldots + S_{m_k}$. Suppose $Y \in B_l$, where $0 \leq l \leq k$. Two cases have to be considered.

Case 1. $l = 0$.

In this case $ip(X, Y)$ is constructed within B_0. Since $t-1 > g \geq m_0$, we have $b_g > \alpha$ or $b_g = \alpha$. If $b_g > \alpha$, $ip(X, Y)$ is constructed as the path that passes the circles C_{t-1}^{α}, C_{t-2}^{α}, ..., C_g^{α}, C_g^{β} in sequence (refer to Figure 5(a), where $\alpha < \beta$ is assumed). Each subpath inside a circle is obtained by executing Vecchia and Sanges's algorithm, and each subpath between two adjacent circles is a (jumping) flipping link. We also note that some of C_{t-1}^{α}, C_{t-2}^{α}, ..., C_g^{α}, C_g^{β} may not exist. For example, if there is a $J_{t-1,t-3}^{\alpha}$ in B_0, then $b_{t-2} = \alpha$. So, C_{t-2}^{α} is not existent and should be removed from the sequence. For all the sequences of circles that appear subsequently, we follow the same convention.

If $b_g = \alpha$, there exists a jumping flipping link, say $J_{u,v}^{\alpha}$, under S_g, where $u > g > v$, and $ip(X, Y)$ is constructed as the path that passes C_{t-1}^{α}, C_{t-2}^{α}, ..., C_u^{α}, C_v^{α}, C_v^{β}, C_{v+1}^{β}, ..., C_g^{β} in sequence (refer to Figure 5(b)).

Case 2. $l > 0$.

In this case, $ip(X, Y)$ contains $l+1$ subpaths that are within B_0, B_1, ..., B_l, respectively, and any adjacent two are joined by a flipping link. Let $(\alpha >) x_1 > x_2 > \ldots > x_n$ be defined as in Algorithm *Stage_Grouping*, i.e., $x_j = \min\{b_{m_{j-1}}, b_{m_{j-1}-1}, \ldots, b_{m_j}\}$ for each $1 \leq j \leq n$ (by Lemma 3.1), where $n = k$ or $k-1$. The subpath within B_0 passes C_{t-1}^{α}, C_{t-2}^{α}, ..., $C_{m_0}^{\alpha}$, and the subpath within B_r passes $C_{m_{r-1}}^{x_r}$, $C_{m_{r-1}-1}^{x_r}$, ..., $C_{m_r}^{x_r}$, where $1 \leq r \leq l-1$. The subpath within B_l is explained below.

If no jumping flipping link exists in B_l ($l = k = n+1$ in this case), the subpath passes $C_{m_{l-1}}^{\beta}$, $C_{m_{l-1}-1}^{\beta}$, ..., C_g^{β}. Otherwise ($l \neq n+1$), let $J_{y_l, z_l}^{x_l}$ be the leftmost (and upmost) jumping flipping link in B_l. Five subcases have to be considered as follows.

Subcase 1. $g \geq y_l$. The subpath passes $C_{m_{l-1}}^{\beta}$, $C_{m_{l-1}-1}^{\beta}$, ..., C_g^{β} (by Lemma 3.1 $b_{m_{l-1}} > b_{m_{l-1}-1} \geq \ldots \geq b_{y_l}$).

Subcase 2. $g=y_l-1$. The subpath passes $C^{\beta}_{m_{l-1}}$, $C^{\beta}_{m_{l-1}-1}$, ..., C^{β}_{g} (by Lemma 3.1 (b_g=)x_l=min{$b_{m_{l-1}}$, $b_{m_{l-1}-1}$, ..., b_{m_l}}).

Subcase 3. $z_l<g<y_l-1$. The subpath passes $C^{x_l}_{m_{l-1}}$, $C^{x_l}_{m_{l-1}-1}$, ..., $C^{x_l}_{y_l}$, $C^{x_l}_{z_l}$, $C^{\beta}_{z_l}$, $C^{\beta}_{z_l+1}$, ..., C^{β}_{g}.

Subcase 4. $g\le z_l$ and $b_g>x_l$. The subpath passes $C^{x_l}_{m_{l-1}}$, $C^{x_l}_{m_{l-1}-1}$, ..., $C^{x_l}_{y_l}$, $C^{x_l}_{z_l}$, ..., $C^{x_l}_{g}$, C^{β}_{g}.

Subcase 5. $g<z_l$ and $b_g=x_l$. By Theorem 2.1 there is a jumping flipping link. say $J^{x_l}_{u,v}$, under S_g (note that $b_{z_l}>x_l$ and $b_{m_l}>x_l$). The subpath passes $C^{x_l}_{m_{l-1}}$, $C^{x_l}_{m_{l-1}-1}$, ..., $C^{x_l}_{y_l}$, $C^{x_l}_{z_l}$, ..., $C^{x_l}_{u}$, $C^{x_l}_{v}$, C^{β}_{v}, C^{β}_{v+1}, ..., C^{β}_{g}.

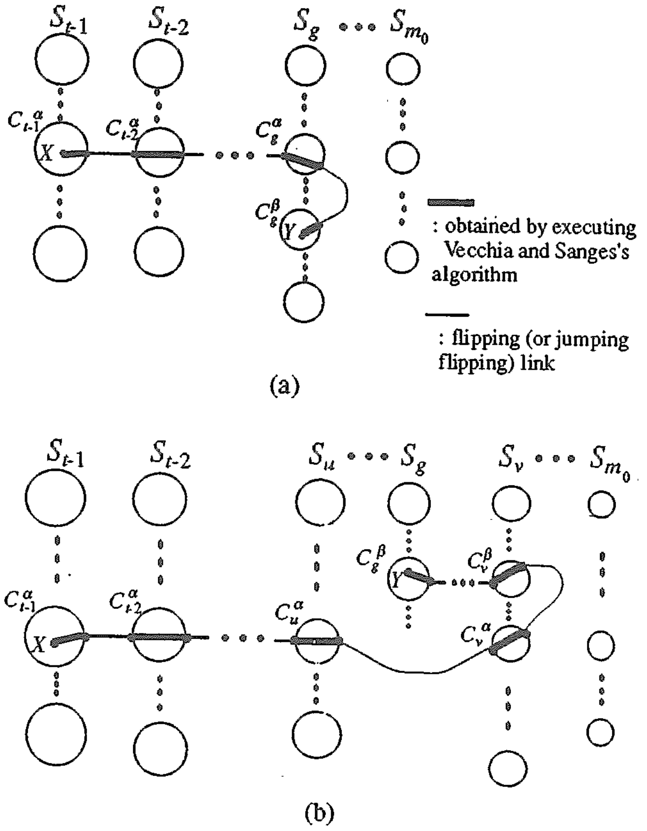

Figure 5. Two examples of $ip(X, Y)$. (a) $b_g>\alpha$. (b) $b_g=\alpha$.

Now we show a more concrete example for the construction of $ip(X, Y)$. Refer to Figure 6. Let X= 3ΔΔΔΔΔΔΔΔΔ and Y=4243410ΔΔΔ be two nodes in IK(5, 10) with coefficient vector (4, 2, 4, 3, 4, 1, 1, 3, 1, $*$), where Δ is a *don't care* symbol. Since $X\in C^{3}_{9}$ and $Y\in C^{0}_{3}$, we have α=3, β=0, and g=3. With (4, 2, 4, 3, 4, 1, 1, 3, 1, $*$) and 3 as input, Algorithm *Stage_Grouping* determines x_1=2, x_2=1, and produces (m_0, m_1, m_2, m_3)=(9, 5, 2, 1). So, $B_0=S_9$, $B_1=S_9+S_8+S_7+S_6+S_5$, $B_2=S_5+S_4+S_3+S_2$, and $B_3=S_2+S_1$. Clearly, Y belongs to B_2 and $ip(X, Y)$ contains three subpaths that are within B_0, B_1, and B_2, respectively. The subpath within B_0 passes C^{3}_{9}. The subpath within B_1 passes C^{2}_{9}, C^{2}_{7}, C^{2}_{6}, and C^{2}_{5}. Since $J^{1}_{5,2}$ is the leftmost jumping flipping link in B_2, the subpath within B_2 passes C^{1}_{5}, C^{1}_{2}, C^{0}_{2}, and C^{0}_{3}. The entire path is shown with darkened lines in Figure 6. For the same example, if $Y\in C^{0}_{4}$, then the subpath in B_2, which is shown with lightened lines in Figure 6, passes C^{0}_{5} and C^{0}_{4}.

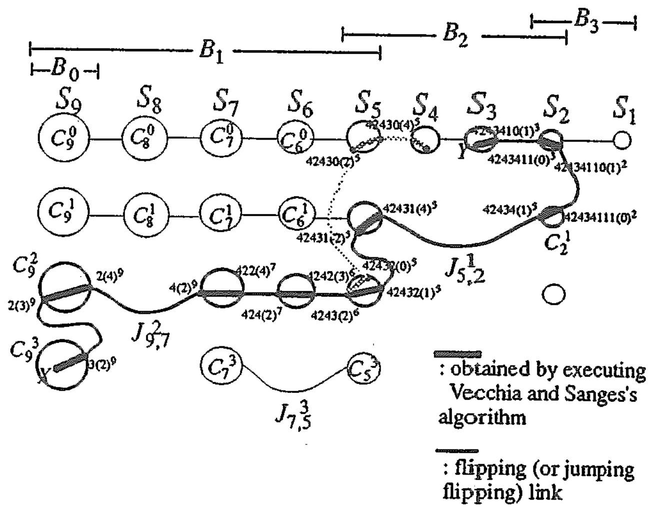

Figure 6. Another example of $ip(X, Y)$.

It is not difficult to see that the length of $ip(X, Y)$ can be determined in $O(t)$ time. Recall that $ip(X, Y)$ can be expressed as an alternate sequence of paths and links, say (P_1, l_1, P_2, l_2, ..., P_c, L_c, P_{c+1}), where $1\le c<2t$. Each l_k, $1\le k\le c$, is a (jumping) flipping link, and each P_q, $1\le q\le c+1$, is a path that is obtained by executing Vecchia and Sanges's algorithm on some embedded K(d, j). Moreover, the two end nodes of P_q are j-frontiers if $2\le q\le c$, and at least one of the two is a j-frontier if q=1 or c+1. By Lemmas 4.1, 4.2, and 4.3, we have the following lemma.

Lemma 4.4. The length of $ip(X, Y)$ can be determined in $O(t)$ time.

Besides, $ip(X, Y)$ is the shortest if X is a t-frontier of S_{t-1}.

Theorem 4.1.[14] If X is a t-frontier of S_{t-1}, then $ip(X, Y)$ is the shortest.

Thus far we have assumed $X\in C^{\alpha}_{f}$, $Y\in C^{\beta}_{g}$, and $f=t-1$. Now we consider the situation of $f<t-1$. If $t-1>f\ge g\ge i$, the routing algorithm is executed on $S_f+S_{f-1}+ ... +S_i$, in spite of

$S_{t-1}+S_{t-2}+ \dots +S_{f+1}$. In this case, $ip(X, Y)$ is entirely contained in $S_f+S_{f-1}+ \dots +S_i$. We note that $S_f+S_{f-1}+ \dots +S_i$ forms an embedded IK(d, f+1) with coefficient vector (b_f, b_{f-1}, ..., b_i, *). On the other hand, if $t-1 \geq g > f \geq i$, $ip(Y, X)$ can be constructed, and so we simply let $ip(X, Y)=ip(Y, X)$, regardless of their directions. The following corollary holds as a consequence of Lemma 4.1 and Theorem 4.1.

Corollary 4.1. If X or Y is a t-frontier of S_{t-1}, their distance can be determined in $O(t)$ time.

It is possible for a t-frontier (at most one) to be positioned outside S_{t-1}. Fortunately, Corollary 4.1 remains true even if X is such a t-frontier.

Corollary 4.2.[14] If X or Y is a t-frontier of IK(d, t), their distance can be determined in $O(t)$ time.

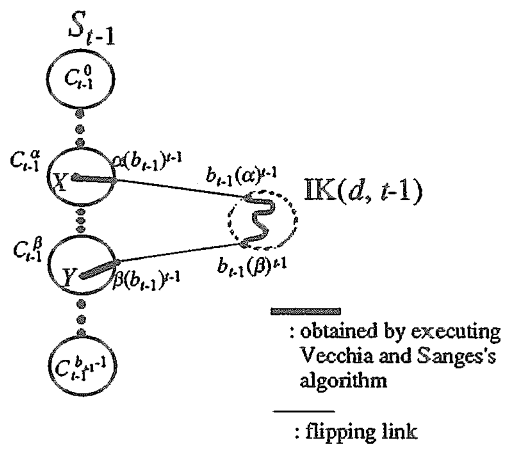

Figure 7. The path that goes through the embedded IK(d, t-1) with coefficient vector (b_{t-2}, b_{t-3}, ..., b_i, *).

5 A shortest-path routing algorithm

In this section, we first show that the shortest path from X to Y can be determined in $O(d \cdot t)$ time. Then we show that it takes only $O(t)$ time for each intermediate node to determine the next node along the shortest path. Suppose the IK(d, t) has coefficient vector (b_{t-1}, b_{t-2}, ..., b_i, *). We first assume $X \in C_{t-1}^{\alpha}$ and $Y \in C_{g}^{\beta}$, where $0 \leq \alpha \leq b_{t-1}-1$, $i \leq g \leq t-1$, and $0 \leq \beta \leq b_g-1$. There are two cases to be discussed below.

Case 1. $g=t-1$. If $\alpha=\beta$ (i.e., X and Y reside in the same embedded K(d, t-1)), Chen and Duh's shortest-path routing algorithm [2] can be applied to determine the shortest path from X to Y in $O(t)$ time. Otherwise ($\alpha \neq \beta$), it takes $O(b_{t-1} \cdot t)$ time to determine the shortest path from X to Y as explained as follows. First, we note that the length of $ip(X, Y)$, which is at most $D_{t-1}+1+D_{t-1}=2^t-1$, serves as an upper bound on the distance between X and Y. Hence, we only need to examine at most b_{t-1} paths: one is $ip(X, Y)$, another goes through the embedded IK(d, t-1) with coefficient vector (b_{t-2}, b_{t-3}, ..., b_i, *) (i.e., $S_{t-2}+S_{t-3}+ \dots +S_i$) (refer to Figure 7), and the others each go through one intermediate embedded K(d, t-1). The path of Figure 7 is not existent if node $b_{t-1}(\alpha)^{t-1}$ or node $b_{t-1}(\beta)^{t-1}$ is not existent. For other paths that go through two or more intermediate embedded K(d, t-1)'s, their lengths are at least $1+D_{t-1}+1+D_{t-1}+1=2^t+1$.

Since the shortest path is concerned, the subpaths inside embedded K(d, t-1)'s and IK(d, t-1) are required to be the shortest. For those subpaths inside embedded K(d,-t-1)'s, their lengths can be determined in $O(t)$ time by Lemmas 4.1, 4.2, and 4.3. On the other hand, if the subpath inside the embedded IK(d, t-1) is existent, its length can be determined in $O(t)$ time by Corollary 4.2. Consequently, we can determine the shortest path fιcm the b_{t-1} paths in $O(b_{t-1} \cdot t)$ time.

Case 2. $g<t-1$. If $b_{t-1} \leq b_{t-2}$, there are b_{t-1} (t-1)-flipping links that connect C_{t-1}^{j} and C_{t-2}^{j} for all $0 \leq j \leq b_{t-1}-1$. Any route from X to Y will traverse one of them. Since a path from X to Y that goes through two intermediate embedded K(d, t-1)'s can be replaced with a shorter path that goes through only one intermediate embedded K(d, t-1), we only need to examine b_{t-1} paths: one does not go through any intermediate embedded K(d, t-1), and the others each go through one intermediate embedded K(d, t-1). The shortest path can be determined from the b_{t-1} paths in $O(b_{t-1} \cdot t)$ time similarly.

If $b_{t-1}>b_{t-2}$, there are b_{t-2} (t-1)-flipping links that connect C_{t-1}^{j} and C_{t-2}^{j} for all $0 \leq j \leq b_{t-2}-1$. Besides, a jumping flipping link that connects $C_{t-1}^{b_{t-2}}$ and $C_{r}^{b_{t-2}}$ may exist, where $i \leq r < t-2$. Any route from X to Y will traverse one of them. Since these links connect S_{t-1} and the embedded IK(d, t-1) formed by $S_{t-2}+S_{t-3}+ \dots +S_i$, their both end nodes are all (t-1)-frontiers. So, the shortest path can be determined in $O((b_{t-2}+1) \cdot t)$ time similarly.

Thus far we have assumed $X \in C_{f}^{\alpha}$, $Y \in C_{g}^{\beta}$, and $f=t-1$. Now we briefly discuss the situation of $f<t-1$. If $t-1>f \geq g \geq i$, we can determine the shortest path within $S_f+S_{f-1}+ \dots +S_i$ for the reason as follows. It has been shown in [13] that $2^t+2^{t-1}-2^i-1$ is a tight upper bound on the diameter of IK(d, t) with coefficient vector (b_{t-1}, b_{t-2}, ..., b_i, *). This imposes an upper bound of $2^{f+1}+2^f-2^i-1$ on the distance between X and Y because $S_f+S_{f-1}+ \dots +S_i$ forms an embedded IK(d,f+1) with coefficient vector (b_f, b_{f-1}, ..., b_i, *). So the shortest path from X to Y is contained in the embedded IK(d,f+1), for otherwise its length is at least $1+D_{f+1}+1+D_{f+1}+1= 2^{f+2}+1>2^{f+1}+2^f-2^i-1$. On the other hand, if $f<g$, the shortest path from X to Y is identical to the shortest path from Y to X, regardless of their directions. The following theorem summarizes our discussion above.

Theorem 5.1. The shortest path between any two nodes of IK(d, t) can be determined in $O(d \cdot t)$ time.

A distributed algorithm can be found in [14], which takes $O(t)$ time for each intermediate node to determine the next node along the shortest path once the shortest path being determined by the source node. The interested readers are encouraged to ask for this report.

Theorem 5.2.[14] With $O(d{\cdot}t)$ time preprocessing performed by the source node, it takes $O(t)$ time for each intermediate node to determine the next node along the shortest path.

6 Concluding Remarks

Generally speaking, shortest-path routing on incomplete networks is more difficult than on corresponding complete networks. For example, there is a shortest-path routing algorithm [1] designed for the star networks, but no shortest-path routing algorithm is available for the incomplete star networks. The shortest-path routing algorithm proposed in [11] is applicable only to the incomplete star networks of size $c{\cdot}k!$, where $1<c\leq k$. Latifi and Bagherzadeh have suspected (see [8]) that designing a shortest-path routing algorithm for the incomplete star networks is difficult, and even if such an algorithm exists, the optimality of the algorithm may not justify its complexity. Besides, no shortest-path routing algorithms for the incomplete rotator graphs and the incomplete WK-recursive networks were designed before.

One contribution of this paper is to demonstrate a useful representation, i.e., the multistage graph representation, for incomplete networks. The multistage graph representation can provide a uniform look at the incomplete WK-recursive networks. By its aid, we have successfully designed an efficient shortest-path routing algorithm for the incomplete WK-recursive networks.

The readers who are interested in the incomplete WK-recursive networks are refered to [13, 15, 16] for more results.

References

[1] S. B. Akers, D. Harel, and B. Krishnamurthy, "The star graph: an attractive alternative to the n-cube," in *Proc. of the Int. Conf. on Parallel Processing*, 1987, pp. 393-400.

[2] G. H. Chen and D. R. Duh, "Topological properties, communication, and computation on WK-recursive networks," *Networks*, vol. 24, no. 6, pp. 303-317, 1994.

[3] P. Corbett, "Rotator graphs: an efficient topology for point-to-point multiprocessor networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, no. 5, pp. 622-626, 1992.

[4] D. R. Duh and G. H. Chen, "Topological properties of WK-recursive networks," *J. of Parallel and Distributed Computing*, vol. 23, no. 3, pp. 468-474, 1994.

[5] K. Fernandes, D. K. Griesen, and A. Kanevsky, "Efficient routing and broadcasting in recursive interconnection networks," in *Proc. of the Int. Conf. on Parallel Processing*, 1994, pp. 51-58.

[6] R. Fernandes, D. K. Friesen, and A. Kanevsky, "Embedding rings in recursive networks," in *Proc. of the Int. Symp. on Parallel and Distributed Processing*, Oct. 1994, pp. 273-280.

[7] H. P. Katseff, "Incomplete hypercubes," *IEEE Trans. on Computers*, vol. C-37, no. 5, pp. 604-608, 1988.

[8] S. Latifi and N. Bagherzadeh, "Incomplete star: an incrementally scalable network based on the star graph," *IEEE Trans. on Parallel and Distributed Systems*, vol. 5, no. 1, pp. 97-102, 1994.

[9] S. Ponnuswamy and V. Chaudhary, "Embedding of cycles in rotator and incomplete rotator graphs," in *Proc. of the Int. Symp. on Parallel and Distributed Processing*, Oct. 1994, pp. 603-610.

[10] F. P. Preparata and J. Vuillemin, "The cube-connected cycles: a versatile network for parallel computation," *Communications of the ACM*, vol. 24, no. 5, pp. 300-309, 1981.

[11] C. P. Ravikumar, A. Kuchlous, and G. Manimaran, "Incomplete star graph: an economical fault-tolerant interconnection network," in *Proc. of the Int. Conf. on Parallel Processing*, vol. 1, 1993, pp. 83-90

[12] M. R. Samatham and D. K. Pradhan, "The de Bruijn multiprocessor networks: a versatile parallel processing and sorting networks for VLSI," *IEEE Trans. on Computers*, vol. 38, no. 4, pp. 567-581, 1989.

[13] M. Y. Su, G. H. Chen, and D. R. Duh, "Topological properties of incomplete WK-recursive networks," in *Proc. of the IEEE Second Int. Conf. on Algorithm & Architecture and Networks*, 1996, pp. 130-137.

[14] M. Y. Su, G. H. Chen, and D. R. Duh, "A shortest-path routing algorithm for the incomplete WK-recursive networks," Technical Report NTUCSIE 95-07, National Taiwan University, Taipei, Taiwan, July 1995.

[15] M. Y. Su, G. H. Chen, and D. R. Duh, "A linear-time algorithm for computing the diameters of incomplete WK-recursive networks," in *Proc. of the Int. Conf. on Parallel and Distributed Systems*, 1996, pp.90-97.

[16] M. Y. Su, G. H. Chen, and D. R. Duh, "Broadcasting on incomplete WK-recursive networks," in *Proc. of the Int. Symp. on Parallel Architectures, Algorithms and Networks*(I-SPAN), 1996, pp.375-381.

[17] G. D. Vecchia and C. Sanges, "A recursively scalable network VLSI implementation," *Future Generation Computer Systems*, vol. 4, no. 3, pp. 235-243, 1988.

[18] G. D. Vecchia and C. Sanges, "An optimized broadcasting technique for WK-recursive topologies," *Future Generation Computer Systems*, vol. 4, no. 3, pp. 353-357, 1989/90.