International Journal of Foundations of Computer Science
Vol. 25, No. 6 (2014) 723–743
© World Scientific Publishing Company
DOI: 10.1142/S0129054114500300

AN ADAPTIVE HEURISTIC ALGORITHM WITH THE PROBABILISTIC SAFETY VECTOR FOR FAULT-TOLERANT ROUTING ON THE (n, k)-STAR GRAPH*

CHIAO-WEI CHIU[†], KUO-SI HUANG[‡],
CHANG-BIAU YANG[†,§] and CHIOU-TING TSENG[†]

[†]*Department of Computer Science and Engineering, National Sun Yat-sen University*
Kaohsiung 80424, Taiwan
[‡]*Department of Information Management, National Kaohsiung Marine University*
Kaohsiung 81157, Taiwan
[§]*cbyang@cse.nsysu.edu.tw*

Received 22 August 2012
Accepted 27 May 2014
Communicated by Koji Nakano

The (n, k)-star graph is a generalization of the n-star graph. It has better scalability than the n-star graph and holds some good properties compared with the hypercube. This paper focuses on the design of the fault-tolerant routing algorithm for the (n, k)-star graph. We adopt the idea of collecting the limited global information used for routing on the n-star graph to the (n, k)-star graph. In the preliminary version of this paper, we built the probabilistic safety vector (PSV) with modified cycle patterns and developed the routing algorithm to decide the fault-free routing path with the help of PSV. Afterwards, we observed that the routing performance of PSV gets worse as the percentage of fault nodes increases, especially it exceeds 25%. In order to improve the routing performance with more faulty nodes, an adaptive method of threshold assignment for the PSV is also proposed. The performance is judged by the average length of routing paths. Compared with distance first search and safety level, PSV with dynamic threshold gets the best performance in the simulations.

Keywords: Interconnection network; (n, k)-star graph; probabilistic safety vector; fault-tolerant routing.

1. Introduction

Interconnection networks play an important role in illustrating the performance of a parallel multi-computer system. The hypercube is one of the famous interconnection networks. The n-star graph [1, 10], denoted as S_n, has been recognized as an alternative to the hypercube. There is a large gap between the number of nodes from S_n to S_{n+1}. The (n, k)-star graph, denoted as $S_{n,k}$, the generalized version of S_n,

*A preliminary version of this paper was presented at the 10th International Symposium on Pervasive Systems, Algorithms and Networks, December, 2009, Kaohsiung, Taiwan.
[§]Corresponding author.

was proposed by Chiang and Chen [4]. $S_{n,k}$ has better scalability and it preserves many attractive properties of S_n, such as *node symmetry*, *distance*, *diameter*, *hierarchical structure*, and *fault-free shortest routing* [1, 4, 12]. Recent research results on *broadcasting* [11], *topological properties* [5], *fault-tolerant connectivity* [9, 15] and *weak-vertex-pancyclicity* [3] demonstrate that $S_{n,k}$ is a very powerful network.

In interconnection networks, *routing* refers to the process of selecting a path of *unicast*, which sends a message from a single source to a single destination through some intermediate nodes. For the reason of efficiency, the length of the path decided by the routing algorithm should be as short as possible. A node is said to be *faulty* if it malfunctions. It can be viewed as that this node and its incident edges are removed from the graph. When some nodes are faulty in a graph, the fault-free shortest routing may not be available.

In this paper, we try to design an algorithm to perform routing on a faulty $S_{n,k}$. When node u wants to send a message to destination v or receives from node t a message which is sent to destination v, this routing algorithm will decide which neighbor to route the message through. Then u will send the message to this neighbor. To avoid the routing loop between nodes, we only consider the neighbors that have not been visited. If there exists no available neighbor, u will ignore the message or send the message back to t. If destination v is reachable, the routing path is composed of all nodes that the message travels. The length of the routing path is one of the criteria for measuring the performance of a routing algorithm.

A routing algorithm can make the decision based on one of the three types of information, *local information*, *global information* and *limited global information.* Local information is that each node only knows the states of its neighbors. One can design a greedy algorithm, such as *distance first search* (DFS), based on local information. Global information is to take care of the state of the whole graph. With the global information, the *Floyd-Warshall* algorithm [6] can solve the all-pairs shortest path problem on a graph with time complexity $O(|V|^3)$, where V denotes the node set, and then we can perform the optimal routing for each pair of nodes. However, it is not efficient while the size of the graph is large. Limited global information considers a limited amount of global information. Each node requires a process for collecting the fault information of its neighbors.

The main challenge in this paper is to design a near-optimal routing algorithm on $S_{n,k}$ with low complexity. Several fault-tolerant routing algorithms on S_n have been proposed [2, 8, 14]. It requires some new approaches to record the accessibility of each node to assist us for routing on $S_{n,k}$. A fault-tolerant routing algorithm, based on the modified cycle patterns of $S_{n,k}$ and the probabilistic safety vector (PSV), is proposed in the preliminary version of this paper. With more simulation cases, we observe that the routing performance of the PSV gets worse as the percentage of fault nodes increases, especially it exceeds 25%. The PSV values will decrease dramatically if the percentage of faulty nodes becomes higher. If most PSV values are less than the fixed threshold, the threshold becomes useless for finding better

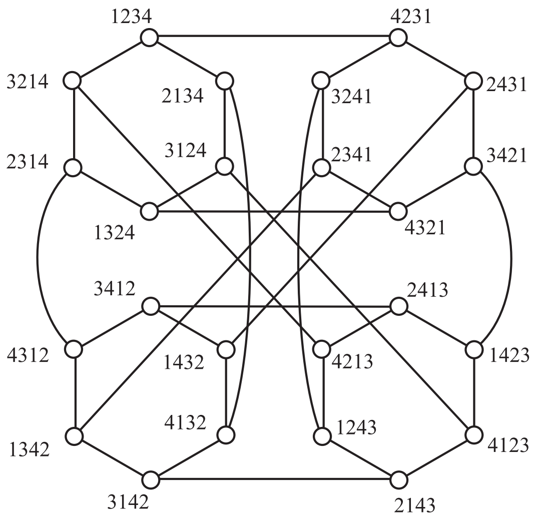


Fig. 1. The diagram of 4-star graph S_4.

routing paths. Thus, an improved PSV algorithm based on the adaptive threshold assignment is proposed in this paper. According to the simulation experiments, the improved PSV algorithm gains better routing lengths than the PSV and other algorithms for various percentages of faulty nodes.

The rest of this paper is organized as follows. In Sec. 2, we give an introduction to the properties of $S_{n,k}$ and a previously proposed routing algorithm for S_n, which can be applied to $S_{n,k}$. Section 3 presents the main body of our method. We extend the probabilistic safety vector of S_n by adding new *cycle patterns* to perform routing on $S_{n,k}$ and propose an adaptive heuristic fault-tolerant routing algorithm to improve the routing performance. In Sec. 4, we show the experimental results of our method and some comparisons with other methods. In Sec. 5, we give the analysis of the probabilistic safety vector. Finally, we give a conclusion in Sec. 6.

2. Preliminaries

2.1. *The n-star and (n,k)-star graphs*

The definition of n-star graph, abbreviated as S_n, was proposed by Akers *et al.* [1]. Each node in S_n is identified by a distinct permutation of the set $\{1, 2, \cdots, n\}$. Nodes $u = u_1u_2\cdots u_n$ and v are adjacent if we can get v by swapping the first symbol of u with another symbol, that is $v = u_iu_2\cdots u_{i-1}u_1u_{i+1}\cdots u_n$. Figure 1 shows S_4 as an example.

The (n,k)-star graph [4], denoted as $S_{n,k}$, is a generalized version of the n-star graph. Similarly, each node of $S_{n,k}$ is identified by a distinct permutation with length k selected from $\{1, 2, \cdots, n\}$, where n and k, $1 \leq k \leq n-1$, are the numbers of available symbols for selection and being selected, respectively. A node u of $S_{n,k}$

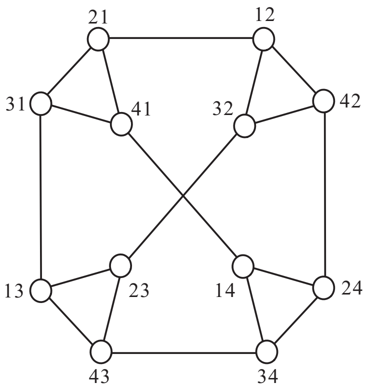


Fig. 2. The diagram of (4,2)-star graph $S_{4,2}$.

is denoted by label $u_1u_2\cdots u_k$, where $u_i \in \{1, 2, \cdots, n\}$ for $1 \le i \le k$, and $u_i \ne u_j$ for $1 \le i < j \le k$. In a node u, the symbols belonging to u are *internal symbols*, and the others are *external symbols*. For example, permutations 7352 and 1583 are two nodes in $S_{8,4}$. In node 7352, internal symbols are 2, 3, 5 and 7; external symbols are 1, 4, 6 and 8. The adjacency in $S_{n,k}$ is more generalized than that in S_n.

In $S_{n,k}$, nodes $u = u_1u_2\cdots u_k$ and v are adjacent if it satisfies one of following cases. Case 1: $v = u_iu_2u_3\cdots\ u_{i-1}\ u_1u_{i+1}\ \cdots u_k$; Case 2: $v = \beta u_2u_3\cdots\ u_k$ where $\beta \in \{1, 2, \cdots, n\} \setminus \{u_1, u_2, \cdots, u_k\}$. In the definition of edges, Case 1 is similar to S_n. In Case 1, since all symbols of u are the same as those of v, edge (u, v) is called an *internal edge*. In Case 2, since the first symbol of v is an external symbol of u, edge (u, v) is called an *external edge*. For example, the diagram of $S_{4,2}$ is shown in Fig. 2. The edge from 42 to 24 is an internal edge, while the edge from 42 to 32 is an external edge.

A graph is called a *regular graph* if the degrees of all nodes are identical. S_n and $S_{n,k}$ are both regular graphs with degree $n-1$. If node u is adjacent to node v, we say that node u is a *neighbor* of node v. For the completeness, we denote the neighbor of node u as $u^{(\alpha)}$, where $\alpha \in \{1, 2, \cdots, n\} \setminus \{u_1\}$. For example, in Fig. 3, consider $u = 4635$ in $S_{7,4}$. The neighbors of u are $u^{(1)} = 1635$, $u^{(2)} = 2635$, $u^{(3)} = 3645$, $u^{(5)} = 5634$, $u^{(6)} = 6435$ and $u^{(7)} = 7635$. Note that for $u = 4635$, there is no $u^{(4)}$ since 4 is the first symbol of u. In this figure, the thin and bold lines represent internal and external edges of u, respectively. According to the definition of $S_{n,k}$, the following properties have been proved [4].

Property 1. [4] *There are $n!$ and $\frac{n!}{(n-k)!}$ nodes in S_n and $S_{n,k}$, respectively. $S_{n,k}$ has better scalability than S_n.*

Property 2. [4] *$S_{n,1}$ is isomorphic to the complete graph, n-clique, and $S_{n,n-1}$ is isomorphic to S_n.*

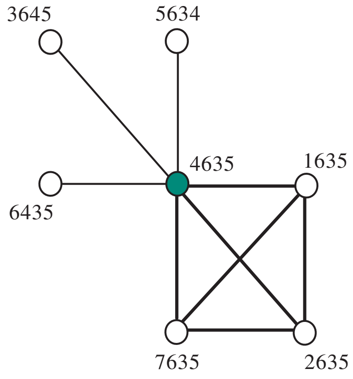


Fig. 3. Neighbors of $u = 4635$ in $S_{7,4}$.

2.2. *Permutation cycles of (n, k)-star graphs*

In $S_{n,k}$, one path can be represented by one or more *cycles* [4]. For source $u = u_1u_2\cdots u_k$ and destination v, a cycle is defined as $\gamma = \{s_1, s_2, \cdots, s_l\}$, where $\gamma \subset \{u_i|1 \leq i \leq k\}$ and l is the length of the cycle. If all symbols in γ are internal symbols of v, we say γ is an *internal cycle* and denote it with a pair of parentheses, $(s_1s_2\cdots s_l)$. If γ has an external symbol of v, we put it as the last symbol s_l, and we put a *desired symbol* t after s_l, where t belongs to v and it will be placed at the position of symbol s_1 of u. Such a γ is called an *external cycle*, and it is denoted by a pair of angle brackets, $\langle s_1s_2\cdots s_l, t\rangle$. Note that there is at most one external symbol in a cycle.

For example, consider source $u = 21745$ and destination $v = 14526$ in $S_{7,5}$, the cycles are $\binom{14526}{21745} = (241)\langle 57, 6\rangle = \langle 57, 6\rangle(241)$. The cycle notations can be explained as follows. The destination of symbol 2 now is occupied by symbol 4. Besides, the destinations of 4 and 1 are occupied by 1 and 2, respectively. Thus, (241) forms an internal cycle. Similarly, the destinations of 5 and 6 are occupied by 7 and 5, respectively. Here, 7 is an external symbol of destination node v and 6 is the desired symbol. So, this external cycle is represented by $\langle 57, 6\rangle$.

As another example, from 2735914 to 8596321 in $S_{9,7}$, the cycles are $\binom{8596321}{2735914} = (39)\langle 57, 6\rangle\langle 214, 8\rangle$.

The *fault-free shortest routing* rule follows the order of the symbols in the cycles to route the message. For example, in $S_{7,5}$, two of the shortest paths from 21745 to 14526 are given as follows:

Path 1: $21745 \rightarrow 41725 \rightarrow 14725 \rightarrow 54721 \rightarrow 74521 \Rightarrow 64521 \rightarrow 14526$
Path 2: $21745 \rightarrow 51742 \rightarrow 71542 \Rightarrow 61542 \rightarrow 21546 \rightarrow 41526 \rightarrow 14526$.

As one can see, the first two routing steps in path 1 and the last two routing steps in path 2 correspond to the internal cycle (241), and the remaining routing steps correspond to the external cycle $\langle 57, 6\rangle$. It is clear that the path length from 21745 to 14526 is 6.

Chiang and Chen [4] proposed a method for generating permutation cycles from the smallest-indexed source I_k=$123 \cdots k$ to any destination v. With slight modification, we can generate the permutation cycles from an arbitrary source u to any destination v in $S_{n,k}$, as described in Algorithm 1.

Algorithm 1: Permutation cycle generation from source u to destination v in $S_{n,k}$.

```
Input: Source $u = u_1u_2 \cdots u_k$ and destination $v = v_1v_2 \cdots v_k$ in $S_{n,k}$
Output: Internal and external cycles from $u$ to $v$
begin
    for $i \leftarrow 1$ to $n$ do
        $UV[i] \leftarrow null$; $VU[i] \leftarrow null$; $S[i] \leftarrow false$;
    for $i \leftarrow 1$ to $k$ do
        $UV[u_i] \leftarrow v_i$; $VU[v_i] \leftarrow u_i$;
    foreach $u_i$ such that $S[u_i] = false$ do
        $CX \leftarrow \{\}$; $CY \leftarrow \{\}$; $doNextStage \leftarrow true$;
        $cur \leftarrow u_i$; $next \leftarrow UV[cur]$; $S[cur] \leftarrow true$; $CX$.append($cur$);
        if $next = cur$ then
            $next \leftarrow null$; $doNextStage \leftarrow false$;
            Output ($CX$) as an internal cycle;
        while $next \neq null$ do
            $cur \leftarrow next$; $next \leftarrow UV[cur]$; $S[cur] \leftarrow true$;
            if $next = null$ then $CY$.append($cur$) ;
            else
                $CX$.append($cur$);
                if $S[next] = true$ then
                    $next \leftarrow null$; $doNextStage \leftarrow false$;
                    Output ($CX$) as an internal cycle;
        if $doNextStage = true$ then
            $CX \leftarrow \{\}$; $next \leftarrow VU[cur]$;
            while $next \neq null$ do
                $cur \leftarrow next$; $next \leftarrow VU[cur]$; $S[cur] \leftarrow true$;
                $CX$.append($cur$);
            Output ($CX$,$CY$) as an external cycle;
```

2.3. *The distance computation of (n, k)-star graphs*

The *distance* between two nodes u and v, denoted as $d(u, v)$, is the length of the shortest path from u to v. The distance of two nodes in a fault-free $S_{n,k}$ is

Table 1. The comparison of S_n and $S_{n,k}$.

Graph	Size	Degree	Diameter
$S_{4,2}$	12	3	3
$S_{5,2}$	20	4	3
$S_4 = S_{4,3}$	24	3	4
$S_{6,2}$	30	5	3
$S_{5,3}$	60	4	5
$S_5 = S_{5,4}$	120	4	6
$S_{6,3}$	120	5	5
$S_{6,4}$	360	5	6
$S_6 = S_{6,5}$	720	5	7
$S_{n,k}$, $1 \le k \le \lfloor \frac{n}{2} \rfloor$	$\frac{n!}{(n-k)!}$	$n-1$	$2k-1$
$S_{n,k}$, $\lfloor \frac{n}{2} \rfloor + 1 \le k \le n-2$	$\frac{n!}{(n-k)!}$	$n-1$	$k + \lfloor \frac{n-1}{2} \rfloor$
$S_n = S_{n,n-1}$	$n!$	$n-1$	$\lfloor \frac{3n-3}{2} \rfloor$

computable. The computation of $d(u, v)$ is given as follows [4]:

$$d(u,v) = \begin{cases} c + m + e, & \text{if } u_1 = v_1; \\ c + m + e - 2, & \text{if } u_1 \neq v_1, \end{cases} \tag{1}$$

where c is the number of cycles, m is the number of misplaced symbols, and e equals 0 or 1 if there exists no external cycle or otherwise, respectively. Note that u_1 and v_1 are the first symbols of u and v, respectively.

For example, in $S_{7,5}$, from 21745 to 14526, $\binom{14526}{21745} = (241)\langle 57, 6\rangle$, we get $c = 2$, $m = 5$, $e = 1$, and $u_1 \neq v_1$. Then we get $d(21745, 14526) = 6$ by Eq. (1). Note that the distance of two nodes in a fault-free S_n can also be computed by Eq. (1) with $e = 0$.

The longest distance for all pairs of nodes in $S_{n,k}$ is called the *diameter* of $S_{n,k}$, which is denoted as $D(S_{n,k})$ and it can be computed as follows [4]:

$$D(S_{n,k}) = \begin{cases} 2k - 1, & \text{if } 1 \le k \le \lfloor \frac{n}{2} \rfloor; \\ k + \lfloor \frac{n-1}{2} \rfloor, & \text{if } \lfloor \frac{n}{2} \rfloor + 1 \le k \le n - 1. \end{cases} \tag{2}$$

By Property 2, diameter $D(S_n) = D(S_{n,n-1})$.

For routing from node u to node v in a fault-free $S_{n,k}$, the neighbors of u can be classified into three kinds as follows. The neighbors of u on the shortest paths to v are called *preferred neighbors*. The distance from one preferred neighbor of u to v is $d(u, v) - 1$. The neighbors of u with the same distance to v are called *semi-preferred neighbors*. The neighbors of u with distance $d(u, v) + 1$ to v are called *non-preferred neighbors*. For example, in Fig. 2, for routing from node 14 to node 23, node 34 is a preferred neighbor, nodes 41 and 24 are semi-preferred neighbors; for routing from node 12 to node 31, node 21 is a preferred neighbor, nodes 32 and 42 are non-preferred neighbors. Table 1 shows the comparison of S_n and $S_{n,k}$ for the properties of size, degree and diameter, where size means the number of nodes contained in the graph.

2.4. *Routing on $S_{n,k}$ with distance first search (DFS)*

For routing a message from node u to node v in $S_{n,k}$ with some faults, because the distance of u and v in a fault-free $S_{n,k}$ can be computed immediately by Eq. (1), the heuristic is to route the message through the nearest neighbor of v. With the local information of the neighbors, we can design a greedy *distance first search* (DFS) algorithm to perform the routing task for each node in $S_{n,k}$. In this DFS algorithm, a non-visited neighbor is selected according to its distance to destination. That is, the neighbor selection rule will follow the priority order: preferred neighbors, semi-preferred neighbors, and non-preferred neighbors.

2.5. *Routing on $S_{n,k}$ with safety levels (SL)*

In an interconnection network with some faults, suppose that each node is allowed to communicate with its neighbors before routing, and it can collect the limited global information as its routing ability. The *safety level* can be used to represent the routing ability of each node in a hypercube [13]. With the same concept, the safety level can also be applied to performing routing on S_n [14]. The safety level of a node u, denoted as $\sigma(u)$, ensures that every node v, from u, with distance less than or equal to $\sigma(u)$ can be reached via the shortest path. It can be computed iteratively as follows [14]:

For i from 1 to diameter $D(S_n)$, compute

$$\sigma(u) = \begin{cases} 0, & \text{if } u \text{ is faulty;} \\ min\{\sigma(u^{(\alpha)})|\alpha \neq u_1\} + 1, & \text{otherwise.} \end{cases} \tag{3}$$

The routing algorithm [14] for a node $u \in S_n$ based on its safety level is to determine whether its neighbors can provide the shortest path to the final destination. If there exists a neighbor that guarantees the shortest path, this neighbor will be selected; otherwise we select the neighbor with the least distance to the destination and the maximal safety level. For performance comparison, we also implement the concept of routing ability with safety levels in $S_{n,k}$.

3. Fault-Tolerant Routing on the (n, k)-Star Graph

3.1. *The modified cycle pattern*

As mentioned in Sec. 2.2, the routing paths from a source node to a destination in $S_{n,k}$ can be represented by one or more *cycles* [4]. In S_n, Yeh *et al.* defined a *cycle pattern* for containing a class of permutation cycles so that the routing paths can be categorized into various cycle patterns [14]. However, the cycle patterns in $S_{n,k}$ are different from that in S_n. The original representation used in S_n is not enough to represent the external cycles in $S_{n,k}$. We add a new symbol Y to describe the external cycles in $S_{n,k}$. From a source u to a destination v in $S_{n,k}$, we define a *modified cycle pattern* for containing a class of cycles, which is composed of some internal cycles and some external cycles. A cycle is represented by at most one A, some Xs and some Ys, where A, X, and Y denote the first symbol, any other internal symbol and any other external symbol of the destination v, respectively. The representation is given as follows:

Rule 1: An internal cycle $\gamma_i = (s_1 s_2 \cdots s_l)$ belongs to $(XXX \cdots XA)$ if some s_i, $1 \le i \le l$, is the first symbol of v; it belongs to $(XXX \cdots XX)$ otherwise. Note that a cycle (s_1) of single symbol belongs to (X), which is omissible since the only one symbol is already at its destination position.

Rule 2: An external cycle $\gamma_e = \langle s_1 s_2 \cdots s_l, t\rangle$, belongs to $\langle XXX \cdots XYA\rangle$ if some s_i, $1 \le i \le l$, is the first symbol of v; it belongs to $\langle XXX \cdots XXY\rangle$ otherwise. Note that symbol t is ignored.

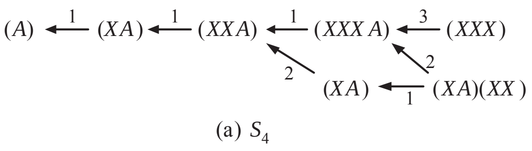
Here, some examples are given for illustration. In $S_{4,2}$, from node 21 to node 32, we have $\binom{32}{21} = \langle 21, 3\rangle$ belongs to $\langle XY\rangle$. $\binom{13}{21} = \langle 12, 3\rangle$ belongs to $\langle YA\rangle$, $\binom{31}{21} = (1)\langle 2, 3\rangle$ belongs to $(X)\langle Y\rangle = \langle Y\rangle$ and $\binom{23}{21} = (2)\langle 1, 3\rangle$ belongs to $(A)\langle Y\rangle$.

In $S_{7,5}$, $\binom{12345}{34561} = (351)\langle 46, 2\rangle$ and $\binom{14526}{21745} = (241)\langle 57, 6\rangle$ both belong to $(XXA)\langle XY\rangle$; $\binom{12345}{72634} = \langle 7, 1\rangle\langle 436, 5\rangle$ belongs to $\langle Y\rangle\langle XXY\rangle$, and $\binom{12345}{43276} = (23)\langle 6, 5\rangle\langle 47, 1\rangle$ belongs to $(XX)\langle Y\rangle\langle XY\rangle$.

Now we present the *connection structure* of $S_{n,k}$, which is a directed graph $G_{n,k}$ with cycle patterns as its vertex set. The definition of *preferred cycle patterns* (PCP) of a cycle pattern is the same as that in S_n [14]. For two cycle patterns P_1 and P_2, if $d(P_2) = d(P_1) - 1$ and there exist two source nodes to the same destination via the shortest path in $S_{n,k}$ which can be represented by P_2 and P_1, then P_2 is called a *preferred cycle pattern* of P_1. This relationship is denoted as $P_2 \in PCP(P_1)$, or alternatively $P_1 \in PCP^{-1}(P_2)$. In $G_{n,k}$, P_1 has a directed edge to P_2 if $P_2 \in PCP(P_1)$.

As examples, the connection structures of cycle patterns in S_4, $S_{4,2}$ and $S_{5,3}$ are shown in Figs. 4 and 5, respectively, where the number aside each arrow describes the number of ways for changing the cycle pattern into its corresponding preferred cycle patterns. For example, $\langle YA\rangle$ and $\langle XY\rangle$ are both preferred cycle patterns of $(A)\langle Y\rangle$, and we have $PCP((A)\langle Y\rangle) = \{\langle YA\rangle, \langle XY\rangle\}$. In fact, Fig. 4(b) is a subgraph of Fig. 5, since $S_{4,2}$ is a subgraph of $S_{5,3}$.

Given a node u and a cycle pattern P, a *preferred neighbor pattern* is denoted as $u^{(\alpha)} \ominus P'$ where $u^{(\alpha)}$ is a neighbor node of u and $P' \in PCP(P)$. Let $\Omega(u, P)$ denote the set of all possible preferred neighbor pattern sets of node u and cycle pattern P. For example, consider node $u = 12$ in $S_{4,2}$ and the connection structure of cycle patterns in Fig. 4(b). The neighbors of node u are $u^{(2)} = 21$, $u^{(3)} = 32$, $u^{(4)} = 42$. Since $PCP((A)\langle Y\rangle) = \{\langle YA\rangle, \langle XY\rangle\}$, we have $\Omega(u, (A)\langle Y\rangle) = \{\{21 \ominus \langle YA\rangle,\ 32 \ominus \langle XY\rangle\},\ \{21 \ominus \langle YA\rangle,\ 42 \ominus \langle XY\rangle\},\ \{32 \ominus \langle YA\rangle,\ 21 \ominus \langle XY\rangle\},\ \{32 \ominus \langle YA\rangle,\ 42 \ominus \langle XY\rangle\},\ \{42 \ominus \langle YA\rangle,\ 21 \ominus \langle XY\rangle\},\ \{42 \ominus \langle YA\rangle,\ 32 \ominus \langle XY\rangle\}\}$. Since $PCP(\langle Y\rangle\langle Y\rangle) = \{\langle XY\rangle\}$, we have $\Omega(u, \langle Y\rangle\langle Y\rangle) = \{\{21 \ominus \langle XY\rangle\},\ \{32 \ominus \langle XY\rangle\},\ \{42 \ominus \langle XY\rangle\}\}$.



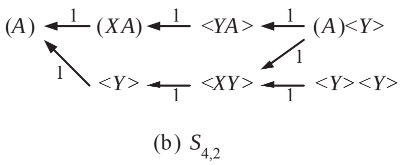


Fig. 4. The connection structures of cycle patterns in (a) S_4 and (b) $S_{4,2}$.

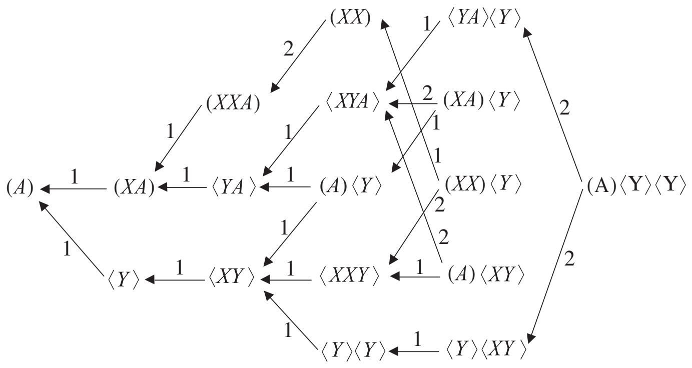


Fig. 5. The connection structure of cycle patterns in $S_{5,3}$.

3.2. *The probabilistic safety vectors*

In $S_{n,k}$, similar to S_n, nodes with the same cycle pattern have similar fault-free shortest paths. The *probabilistic safety vector* (PSV) is used to store the routing abilities of all cycle patterns for each node. Before the routing process is performed in $S_{n,k}$, each node exchanges the PSV with its neighbors iteratively. The value of cycle pattern P in the PSV of node u is denoted as $PSV(u)[P]$, which can be computed by the following steps:

Step 1: Initially,

$$PSV(u)[P] = \begin{cases} 1, & \text{if } P = (A) \text{ and } u \text{ is fault-free;} \\ 0, & \text{if } u \text{ is faulty.} \end{cases}$$

Step 2: For each P, $d(P)$= 1 to $D(S_{n,k})$, compute

$$PSV(u)[P] = \frac{1}{|\Omega(u,P)|} \sum_{S\in\Omega(u,P)} \max\{PSV(u^{(\alpha)})[P'] | u^{(\alpha)} \ominus P' \in S\}.$$

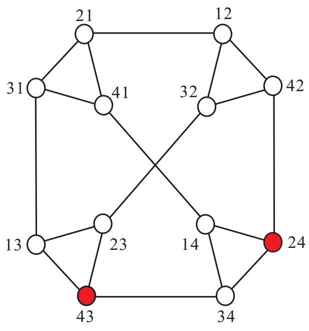


Fig. 6. $S_{4,2}$ with two faulty nodes 24 and 43.

Table 2. Probabilistic safety vectors of Fig. 6.

Label of node	**Probabilistic safety vector**					
	(XA)	$\langle Y\rangle$	$\langle XY\rangle$	$\langle YA\rangle$	$\langle Y\rangle\langle Y\rangle$	$(A)\langle Y\rangle$
12	1	1	0.89	0.89	0.81	0.93
13	0.67	0.67	0.56	0.56	0.48	0.78
14	0.67	0.67	0.44	0.44	0.37	0.67
21	1	1	1	1	0.89	0.89
23	0.67	0.67	0.56	0.56	0.44	0.7
24	0	0	0	0	0	0
31	1	1	0.89	0.89	0.81	0.96
32	1	1	0.78	0.78	0.7	0.81
34	0.33	0.33	0.22	0.22	0.15	0.3
41	1	1	0.89	0.89	0.78	0.96
42	0.67	0.67	0.67	0.67	0.56	0.85
43	0	0	0	0	0	0

As examples, Table 2 shows the PSV of each node in $S_{4,2}$ with two faulty nodes, 24 and 43, as shown in Fig. 6. For example, suppose node 21 is a source for routing. Since $\binom{32}{21} = \langle 21,3\rangle$ and $\binom{42}{21} = \langle 21,4\rangle$, both nodes 32 and 42 belongs to $\langle XY\rangle$. The probability for routing a message from node 21 to node 32 or 42 via the shortest path is 1. Since $\binom{23}{21} = (2)\langle 1,3\rangle$ belongs to $(A)\langle Y\rangle$, the probability for routing a message from 21 to 23 via the shortest path is very high (0.89).

3.3. *Routing with probabilistic safety vectors*

In S_n, one node may have two kinds of neighbors, preferred and non-preferred. However, in $S_{n,k}$, one more kind of neighbor, semi-preferred, exists. Thus, the routing algorithm with PSV for $S_{n,k}$ has to be redesigned. In Algorithm 2, we first guarantee the optimal routing path by checking the routing abilities of cycle patterns in preferred neighbors, and then check the semi-preferred and the non-preferred neighbors sequentially. Then, since nothing can be sure in the final situation, we select the neighbor with the shortest distance and the best routing ability as a heuristic.

For example, consider the routing from node 14 to node 23 in Fig. 6, and suppose that the threshold $\theta = 0.5$. The process of the PSV routing is shown in Table 3. The neighbors of node 14 are nodes 34, 41, and 24, which are the preferred neighbor, semi-preferred neighbor, and a faulty node, respectively. $P(34,23) = \langle XY\rangle, P(41,23) = \langle Y\rangle\langle Y\rangle$. First, we check $PSV(34)[\langle XY\rangle] = 0.22$, which is less than θ and then it is not selected. Second, we check $PSV(41)[\langle Y\rangle\langle Y\rangle] = 0.78$, which is greater than θ, so we route the message to node 41. In Table 3, the neighbor marked with an underline is chosen by the PSV routing algorithm. Collecting the chosen neighbors, we obtain the routing path (14, 41, 31, 13, 23), whose length is 4.

4. Experimental Results

4.1. *Simulations with fixed threshold*

We provide the simulations of routing ability in some (n, k)-star graphs for the three methods, distance first search (DFS), safety level (SL) and probabilistic safety vector (PSV), and then compare the results with the optimal solution (OPT) obtained by

Algorithm 2: PSV routing algorithm on $S_{n,k}$.

```
Input: Source $u$, destination $v$, threshold $\theta$, and PSV
Output: $u^{(\alpha)}$
begin
    if $u = v$ then
        Stop.
    else if $\exists u^{(\alpha)} \in \{preferred\ neighbors\} \setminus \{visited\ neighbors\}$ such that
    $PSV(u^{(\alpha)})[P(u^{(\alpha)}, v)] \geq \theta$ then
        return $u^{(\alpha)}$ ;                     /* Check optimal paths */
    else if $\exists u^{(\alpha)} \in \{semi\text{-}preferred\ neighbors\} \setminus \{visited\ neighbors\}$ such that
    $PSV(u^{(\alpha)})[P(u^{(\alpha)}, v)] \geq \theta$ then
        return $u^{(\alpha)}$ ;                  /* Check suboptimal paths */
    else if $\exists u^{(\alpha)} \in \{non\text{-}preferred\ neighbors\} \setminus \{visited\ neighbors\}$ such that
    $PSV(u^{(\alpha)})[P(u^{(\alpha)}, v)] \geq \theta$ and $P(u^{(\alpha)}, v) \neq (X \cdots XA)$ then
        return $u^{(\alpha)}$ ;   /* There is no shorter path via $(X \cdots XA)$ */
    else if $\exists u^{(\alpha)} \in \{preferred\ neighbors\} \setminus \{visited\ neighbors\}$ such that
    $PSV(u^{(\alpha)})[P(u^{(\alpha)}, v)]$ is maximum then
        return $u^{(\alpha)}$
    else if $\exists u^{(\alpha)} \in \{semi\text{-}preferred\ neighbors\} \setminus \{visited\ neighbors\}$ such that
    $PSV(u^{(\alpha)})[P(u^{(\alpha)}, v)]$ is maximum then
        return $u^{(\alpha)}$
    else if $\exists u^{(\alpha)} \in \{non\text{-}preferred\ neighbors\} \setminus \{visited\ neighbors\}$ such that
    $PSV(u^{(\alpha)})[P(u^{(\alpha)}, v)]$ is maximum then
        return $u^{(\alpha)}$
```

Table 3. The PSV routing path from node 14 to node 23 of Fig. 6, where the underlined neighbor is selected.

Label of node	**Non-visited neighbors**					
	preferred		semi-preferred		non-preferred	
14	34	$\langle XY\rangle$	$\underline{41}$, 24	$\langle Y\rangle\langle Y\rangle$		
41	$\underline{31}$	$\langle XY\rangle$	21	$\langle Y\rangle\langle Y\rangle$		
31	$\underline{13}$	$\langle Y\rangle$			21	$\langle Y\rangle\langle Y\rangle$
13	$\underline{23}$	$\langle A\rangle$	43	$\langle Y\rangle$		

using the Floyd-Warshall algorithm [6], whose time complexity is $O(|V|^3)$, where V denotes the node set in $S_{n,k}$. Note that the Floyd-Warshall algorithm can always find the shortest path between every pair of nodes, but it needs the global information of the graph.

In the simulation program, we build a set of node objects to constitute $S_{n,k}$. Each node object contains data fields for storing node label, pointers to its neighboring nodes with their edge types and data vectors of SL and PSV. We can perform the simulation experiments by setting various scale of faulty nodes randomly. We first set the threshold $\theta = 0.5$ for the PSV value, which means the probability of successful routing with the fault-free shortest path is approximately 0.5. The simulation is performed on a PC with 3.4GHz CPU and 4GB memory.

In the first experiment, we focus on $S_{5,3}$, which contains 60 nodes. We consider the cases of 0, 5, 10, 15, 20, 25 and 30 faulty nodes, and randomly generate 500 graphs for each case. We discard the unreachable paths in this simulation. The result is shown in Table 4(a). For each graph, the average routing length of method M between every pair of nodes is recorded, which is denoted as $L(M)$. The error rate $\frac{L(M)-L(OPT)}{L(OPT)}$ is shown inside the parenthesis following $L(M)$, which makes it easier for us to compare the routing abilities of these methods.

The second experiment considers $S_{6,3}$, which contains 120 nodes. We consider the cases that the number of faulty nodes varies from 0 to 40 with step 5, and randomly generate 200 graphs for each case. The result is shown in Table 4(b). When the percentage of faulty nodes is less than 12% in $S_{6,3}$, $L(PSV)$ is very close to $L(OPT)$.

In the third experiment, we consider $S_{7,4}$, which contains 840 nodes. We consider the cases of 0, 20, 40, 60, 80, 100, 150, 200 and 250 faulty nodes, and randomly generate 100 graphs for each case. For each graph with method M, we randomly select 1000 pairs of nodes, and then compute the average length $L(M)$ of reachable paths. The result is shown in Table 4(c). Although the size of $S_{7,4}$ is large, PSV still gets better result than DFS and SL.

Table 5 shows the comparison of the average preprocessing time of each node to gather the routing information and the average routing time from u to v of each node-pair (u, v) of various methods. We can observe that the PSV method requires

Table 4. Comparison of average routing length of various methods, where threshold $\theta = 0.5$.

(a) $S_{5,3}$

Number of faulty nodes	**Average routing length**			
	OPT	DFS	SL	PSV
0 (0%)	3.136	3.136 (0%)	3.136 (0%)	3.136 (0%)
5 (8%)	3.304	3.650 (10%)	3.492 (5%)	3.399 (2%)
10 (16%)	3.514	4.339 (23%)	4.047 (15%)	3.859 (9%)
15 (25%)	3.810	5.423 (42%)	5.085 (33%)	4.706 (23%)
20 (33%)	4.206	6.926 (64%)	6.727 (59%)	6.065 (44%)
25 (41%)	4.697	8.727 (85%)	8.610 (83%)	7.926 (68%)
30 (50%)	4.661	8.139 (74%)	8.067 (73%)	7.504 (61%)

(b) $S_{6,3}$

Number of faulty nodes	**Average routing length**			
	OPT	DFS	SL	PSV
0 (0%)	3.429	3.429 (0%)	3.429 (0%)	3.429 (0%)
5 (4%)	3.513	3.719 (5%)	3.624 (3%)	3.579 (1%)
10 (8%)	3.602	4.051 (12%)	3.890 (7%)	3.778 (4%)
15 (12%)	3.698	4.434 (19%)	4.205 (13%)	4.021 (8%)
20 (16%)	3.814	4.890 (28%)	4.628 (21%)	4.373 (14%)
25 (20%)	3.945	5.464 (38%)	5.139 (30%)	4.835 (22%)
30 (25%)	4.113	6.208 (50%)	5.915 (43%)	5.519 (34%)
35 (29%)	4.273	6.968 (63%)	6.696 (56%)	6.262 (46%)
40 (33%)	4.520	8.279 (83%)	8.075 (78%)	7.784 (72%)

(c) $S_{7,4}$

Number of faulty nodes	**Average routing length**			
	OPT	DFS	SL	PSV
0 (0%)	4.585	4.585 (0%)	4.585 (0%)	4.585 (0%)
20 (2%)	4.650	4.985 (7%)	4.792 (3%)	4.706 (1%)
40 (4%)	4.711	5.351 (13%)	5.046 (7%)	4.843 (2%)
60 (7%)	4.778	5.667 (18%)	5.304 (10%)	4.994 (4%)
80 (9%)	4.825	6.065 (25%)	5.600 (16%)	5.178 (7%)
100 (11%)	4.922	6.501 (32%)	5.994 (21%)	5.404 (9%)
150 (17%)	5.048	7.279 (44%)	6.751 (33%)	5.936 (17%)
200 (23%)	5.271	9.025 (71%)	8.420 (59%)	7.139 (35%)
250 (29%)	5.552	10.705 (92%)	10.261 (84%)	9.704 (74%)

more time for checking more status than other methods. Hence it can obtain smaller average routing length than DFS and SL methods.

4.2. *Dynamic threshold assignment for the PSV*

In the experimental results in Sec. 4.1, we can see that the result of PSV is not so good when the percentage of faulty nodes exceeds 25%. When the number of

Table 5. Comparison of average preprocessing time of each node and average routing time of each routing node-pair of various methods for $S_{5,3}$, $S_{6,3}$ and $S_{7,4}$ in millisecond (ms), where threshold $\theta = 0.5$.

(a) $S_{5,3}$

Number of faulty nodes	**Average preprocessing time**				**Average routing time**			
	OPT	DFS	SL	PSV	OPT	DFS	SL	PSV
0(0%)	4.75	0	< 0.01	3.63	0	2.05	1.98	8.57
15(25%)	3.90	0	< 0.01	5.48	0.07	4.56	5.25	15.66
30(50%)	4.22	0	< 0.01	3.58	0.11	4.46	7.14	43.15

(b) $S_{6,3}$

Number of faulty nodes	**Average preprocessing time**				**Average routing time**			
	OPT	DFS	SL	PSV	OPT	DFS	SL	PSV
0(0%)	6.91	0	< 0.01	1.81	0.04	2.89	2.89	11.44
20(16%)	5.70	0	< 0.01	2.61	0	6.27	8.66	25.08
40(33%)	5.46	0	< 0.01	2.57	0.06	11.82	13.78	166.64

(c) $S_{7,4}$

Number of faulty nodes	**Average preprocessing time**				**Average routing time**			
	OPT	DFS	SL	PSV	OPT	DFS	SL	PSV
0(0%)	162.54	0	0.06	9.34	0	6.76	6.27	23.87
80(9%)	154.01	0	< 0.01	10.66	0.2	9.77	14.17	24.47
150(17%)	146.05	0	< 0.01	10.14	0	12.42	16.25	35.62
250(29%)	135.82	0	0.02	9.06	0	11.37	27.65	153.76

faulty nodes grows, the values in probabilistic safety vectors will decrease, and the threshold $\theta = 0.5$ in the routing algorithm is no longer suitable. After a series of experiments were done for trying good thresholds, we find that the trend of the threshold with the best result is similar to a decreasing function.

For example, in the $(6,3)$-star graph, we provide the comparison of the PSV algorithm with the thresholds 0.5, 0.45, 0.4, 0.35, 0.3, 0.25, 0.2, 0.15, 0.1 and 0.05 in Table 6. We randomly generate 50 graphs for each number of faulty nodes. For each case of the threshold, we randomly select 1000 pairs of nodes, and then compute the average length $L(PSV)$ of reachable paths. In Table 6, the performances with different thresholds are comparable under low percentages of faulty nodes. We observe that the better thresholds are about 0.25, 0.2 and 0.2 while the percentages of faulty nodes are 29%, 41% and 45%, respectively. The trend of the threshold with the best result is approximately proportional to $0.33 \times \frac{|V \setminus F|}{|V|}$, where V and F denote the sets of all nodes and faulty nodes, respectively.

Thus, we define the following formula for deciding the threshold for each $S_{n,k}$,

$$\theta = \lambda \times \frac{|V \setminus F|}{|V|}, \tag{4}$$

Table 6. The comparison of PSV with various thresholds on $S_{6,3}$.

# of		$\frac{L(PSV)-L(OPT)}{L(OPT)}$ **with threshold:**									
faults	$L(OPT)$	0.5	0.45	0.4	0.35	0.3	0.25	0.2	0.15	0.1	0.05
0(0%)	3.429	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
5(4%)	3.514	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%
10(8%)	3.604	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%
15(12%)	3.700	9%	9%	9%	9%	9%	9%	9%	9%	9%	9%
20(16%)	3.817	15%	14%	14%	14%	14%	14%	14%	14%	14%	14%
25(20%)	3.952	24%	23%	23%	23%	23%	23%	24%	24%	24%	24%
30(25%)	4.093	34%	33%	31%	30%	30%	31%	31%	31%	31%	31%
35(29%)	4.229	45%	43%	37%	36%	36%	36%	36%	37%	37%	37%
40(33%)	4.511	75%	72%	57%	54%	55%	54%	54%	55%	55%	56%
45(37%)	4.729	94%	85%	68%	66%	66%	65%	65%	68%	69%	69%
50(41%)	5.154	146%	143%	109%	105%	100%	100%	97%	100%	101%	102%
55(45%)	5.471	188%	185%	136%	131%	133%	127%	125%	128%	129%	129%
60(50%)	5.810	219%	212%	165%	157%	151%	151%	147%	148%	150%	151%

where λ is a constant less than 0.5, V and F are the sets of all nodes and faulty nodes, respectively. The degree and diameter of $S_{n,k}$ are proportional to the values of n and k. If the percentage of faulty nodes is large, the routing probability decreases while the routing length or diameter increases. And a node of large degree has better chance to find a good routing path. A heuristic is to set the constant λ to $0.5 \times \frac{n}{n+k}$. We are not sure whether the threshold decided by the formula is the best, but we find that it is better than the fixed threshold 0.5 based on the simulation experiments. The PSV algorithm with dynamic threshold is denoted as PSV+.

We perform experiments of the five methods, OPT, DFS, SL, PSV and PSV+, on $S_{6,4}$, which contains 360 nodes. The constant λ is set to 0.3 in the experiments here. For each method, we consider the cases that the number of faulty nodes varies from 0 to 120 with step 10, and we randomly generate 20 graphs for each number of faulty nodes. For each graph, we randomly select 1000 pairs of nodes, and record the average routing length of the optimal solution, $L(OPT)$, and each method, $L(M)$. Then we plot the average routing length of each method, as shown in Fig. 7. We can see that there is a large improvement from PSV to PSV+ when the number of faulty nodes is more than 90 (25%).

We perform another experiments of the five methods, OPT, DFS, SL, PSV and PSV+, on $S_{8,5}$, which contains 6720 nodes. The constant λ is set to 0.307 in these experiments. We consider the cases that the number of faulty nodes varies from 0 to 3000 with step 100, and randomly generate 10 graphs for each number of faulty nodes. For each graph, we randomly select 1000 pairs of nodes, and plot the average routing length of the optimal method (OPT), DFS, SL, PSV and PSV+, as shown in Fig. 8.

In Figs. 7 and 8, we can see that the safety level does not work well when the number of faulty nodes grows. The neighbor decided by distance first search (DFS) is even better than that based on the safety level. The probabilities of PSV decrease as the number of faulty nodes increases. The potentially optimal path with low PSV

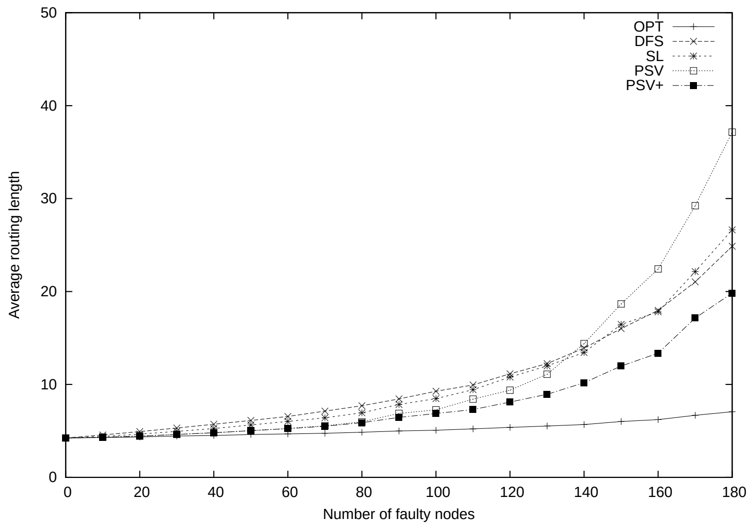


Fig. 7. The average routing length of each method on $S_{6,4}$, which contains 360 nodes.

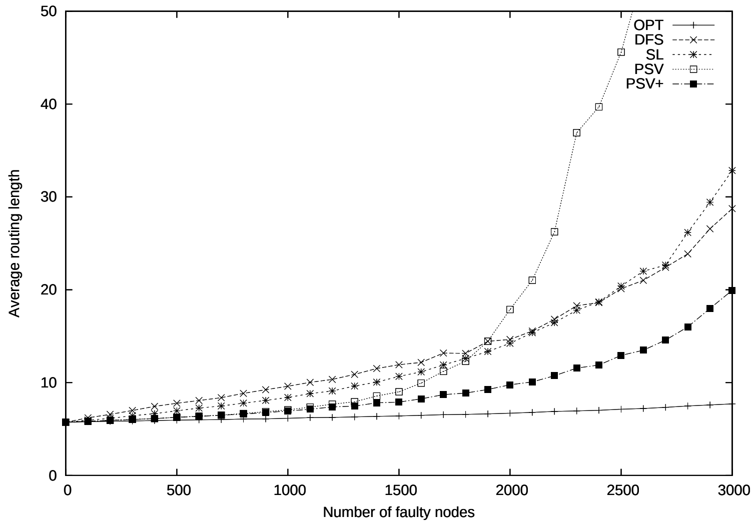


Fig. 8. The average routing length of each method on $S_{8,5}$, which contains 6720 nodes.

will be ignored by the fixed $\theta = 0.5$ in the routing algorithm. It can observed that the routing performance of PSV gets worse as the percentage of fault nodes increases, especially it exceeds 25%. However, the probabilistic safety vector still work well after we dynamically assign the threshold for our algorithm.

Table 7 shows the comparison of the average preprocessing time of each node to gather the routing information and the average routing time from u to v of each

Table 7. Comparison of average preprocessing time of each node and average routing time of each routing node-pair of various methods for $S_{6,4}$ and $S_{8,5}$ in millisecond (ms), where threshold $\theta = 0.5$ for PSV.

(a) $S_{6,4}$

Number of	**Average preprocessing time**					**Average routing time**				
faulty nodes	OPT	DFS	SL	PSV	PSV+	OPT	DFS	SL	PSV	PSV+
0(0%)	6.33	0	< 0.01	0.74	0.74	0	12.54	6.17	10.13	12.59
90(25%)	5.21	0	< 0.01	0.82	0.82	0	15.33	12.65	37.52	30.66
180(50%)	4.51	0	< 0.01	0.69	0.69	0	21.69	24.63	835.29	146.63

(b) $S_{8,5}$

Number of	**Average preprocessing time**					**Average routing time**				
faulty nodes	OPT	DFS	SL	PSV	PSV+	OPT	DFS	SL	PSV	PSV+
0(0%)	967.80	0	< 0.01	6.97	6.97	0	7.70	4.70	41.81	31.31
1000(14%)	946.78	0	< 0.01	7.21	7.21	0	19.92	33.12	58.57	29.46
2000(29%)	923.59	0	< 0.01	6.18	6.18	0	30.66	56.93	1521.86	95.83
3000(44%)	901.59	0	< 0.01	5.02	5.02	0	32.28	77.91	13578.01	423.86

node-pair (u, v) of various methods. We can observe that the preprocessing time and routing time are proportional to the size of $S_{n,k}$ and the routing length. While the percentage of fault nodes increases especially it exceeds 25%, PSV makes bad decision and the lengths of routing path are also increasing. Then PSV may not find a good routing path under this situation. Instead of the fixed probability threshold, PSV+ applies the adaptive threshold to getting good routing path and to saving the routing time.

5. Analysis of the Probabilistic Safety Vector

Let $\psi(n)$, $\psi(n, k)$ denote the length of the probabilistic safety vector of S_n and $S_{n,k}$, respectively, which describe the number of all distinct cycle patterns. In S_n, $\psi(n)$ can be calculated by the function $\sum_{k=0}^{n} p(k)$ [14], where $p(k)$ is the number of partitions of k [7]. For example, the partitions of 4 are 4, $3+1$, $2+2$, $2+1+1$ and $1+1+1+1$, so $p(4) = 5$. We get $\psi(4) = p(0)+p(1)+p(2)+p(3)+p(4) = 1+1+2+3+5 = 12$.

In $S_{n,k}$, it is hard to calculate $\psi(n, k)$ immediately. In our experiments, we implement a function *computePattern* to check the cycle pattern of a source to each node. In Property 2, $S_{n,1}$ is isomorphic to the complete graph, and $S_{n,n-1}$ is isomorphic to S_n. Thus, we do not have to compute $\psi(n, 1)$ and $\psi(n, n-1)$. $\psi(n, n)$ is given as $\psi(n)$ in S_n. We show some values of $\psi(n, k)$ in Table 8.

We get the following property of the length of the probabilistic safety vector in $S_{n,k}$.

Table 8. The length of the probabilistic safety vector in $S_{n,k}$ $(3 \leq n \leq 15)$.

	The length of probabilistic safety vector						
n	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$
3	4	4					
4	7	7	7				
5	7	17	12	12			
6	7	18	36	19	19		
7	7	18	40	69	30	30	
8	7	18	41	80	124	45	45
9	7	18	41	84	150	212	67
10	7	18	41	85	161	266	349
11	7	18	41	85	165	292	453
12	7	18	41	85	166	303	507
13	7	18	41	85	166	307	533
14	7	18	41	85	166	308	544
15	7	18	41	85	166	308	548

Property 3. *If $n > 2k$, $\psi(n,k)$ is equal to $\psi(2k,k)$.*

The reason is that the number of external symbols is $n-k$. If $n > 2k$, then $n-k > k$, which means the number of possible external cycles has reached the upper bound k, and no more new cycle pattern can be constructed. The sequence of $\psi(2k,k) = \{7, 18, 41, 85, 166, 308, \cdots\}$ is interesting but its general form is still unknown.

A routing algorithm is able to make the decision based on the held information in each node. The Floyd-Warshall (OPT) algorithm can find the optimal routing path but it requires a centralized controller to maintain the global state of this network. The DFS (distance first search) can find the next routing node by calculating the distance from the node labels of destination and neighbors. It is a local information routing strategy and each node does not know the faulty state of other nodes. Hence the OPT and DFS methods keep no information about other nodes. In the preprocessing stage each node will gather the information for routing. Then each node uses the gathered information to decide the next routing node in the routing stage.

Note that each node only gathers the information from its neighbors in SL and PSV methods. Hence LS and PSV methods are based on the limited global information for routing. The time required for requesting the information of neighbors to compute the PSV of each node is $O(n\psi(n,k))$, and the space for storing the PSV for each node is $O(\psi(n,k))$. In our observation, $\psi(n,k)$ grows slowly when the size of $S_{n,k}$ becomes large. The time complexity for our algorithm to route through one path of length L is $O(nL)$, which is the same as that for DFS and SL. The space and time complexities of each node on $S_{n,k}$ is shown in Table 9, where D and V denote the diameter and the node set of $S_{n,k}$, respectively.

Table 9. Space and time complexities for various methods on $S_{n,k}$.

For each node	DFS	SL	PSV	OPT
space requirement	$O(1)$	$O(D)$	$O(\psi(n,k))$	$O(1)$
preprocessing	none	$O(nD)$	$O(n\psi(n,k))$	$O(\|V\|^3)$
routing	$O(n)$	$O(n)$	$O(n)$	$O(1)$

6. Conclusion

In this paper, we apply the idea of collecting the limited global information used for routing on the n-star graph to the (n,k)-star graph. We modify the cycle patterns of star graph S_n, and propose the routing algorithm for $S_{n,k}$ based on the probabilistic safety vector (PSV). This paper improves the performance of routing on $S_{n,k}$ with more faulty nodes. Based on the observation on the routing performance of various faulty scales, we propose a heuristic formula for dynamically assigning threshold values in our algorithm. Compared with the distance first search algorithm (DFS) and the safety level algorithm (SL), the average routing length of our method is the best. In the future, it is worth to discuss the relationship between faulty states and the probabilistic safety vector model. It may guide a better way for the modification of the PSV model to get better routing performance.

Acknowledgments

The authors gratefully acknowledge the helpful suggestions and comments of the reviewers, which have improved the presentation.

References

[1] S. B. Akers, D. Horel and B. Krishnamurthy, The star graph: An attractive alternative to the n-cube, *Proceeding of the International Conference on Parallel Processing* (1987) 393–400.

[2] N. Bagherzadeh, M. Nassif and S. Latifi, A routing and broadcasting scheme on faulty star graphs, *IEEE Transactions on Computers* **42** (Nov. 1993) 1398–1403.

[3] Y.-Y. Chen, D.-R. Duh, T.-L. Ye and J.-S. Fu, Weak-vertex-pancyclicity of (n,k)-star graphs, *Theoretical Computer Science* **396** (2008) 191–199.

[4] W.-K. Chiang and R.-J. Chen, The (n,k)-star graph: A generalized star graph, *Information Processing Letters* **56** (Dec. 1995) 259–264.

[5] W.-K. Chiang and R.-J. Chen, Topological properties of the (n,k)-star graph, *International Journal of Foundations of Computer Science* **9**(2) (1998) 235–248.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, 2nd edn. (MIT Press and McGraw-Hill, 2001).

[7] R. L. Graham, D. E. Knuth and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, 2nd edn. (Addison-Wesley Longman Publishing Co., Inc., 1994).

[8] Q. P. Gu and S. Peng, Node-to-node cluster fault routing in star graph, *Information Processing Letters* **56** (1995) 29–35.

[9] H. Hsu, Y. Hsieh, J. Tan and L. Hsu, Fault hamiltonicity and fault hamiltonian connectivity of the (n, k)-star graphs, *Networks* **42** (Apr. 2003) 189–201.

[10] S.-C. Hu and C.-B. Yang, Fault tolerance on star graphs, *International Journal of Foundations of Computer Science* **8**(2) (1997) 127–142.

[11] J. Li, M. Chen, Y. Xiang and S. Yao, Optimum broadcasting algorithms in (n, k)-star graphs using spanning trees, *IFIP International Federation for Information Processing* (2007) 220–230.

[12] T.-C. Lin and D.-R. Duh, Constructing vertexdisjoint paths in (n, k)-star graphs, *Information Sciences* **178**(3) (2008) 788–801.

[13] J. Wu, Unicasting in faulty hypercubes using safety levels, *IEEE Transactions on Computers* **46** (Feb. 1997) 241–244.

[14] S.-I. Yeh, C.-B. Yang and H.-C. Chen, Fault-tolerant routing on the star graph with safety vectors, *Proc. of the Sixth Annual International Symposium on Parallel Architectures, Algorithms, and Networks, I-SPAN*02, Manila, Philippines (May 2002), pp. 301–306.

[15] A. Yuan, E. Cheng and L. Lipták, Linearly many faults in (n, k)-star graphs, *International Journal of Foundations of Computer Science* **22**(7) (2011) 1729–1745.