Fault-Tolerant Ring Embedding in a Star Graph with Both Link and Node Failures

Yu-Chee Tseng, *Member*, *IEEE* Computer Society, Shu-Hui Chang, and Jang-Ping Sheu, *Member*, *IEEE Computer Society*

Abstract—The star graph interconnection network has been recognized as an attractive alternative to the hypercube network. Previously, the star graph has been shown to contain a Hamiltonian cycle. In this paper, we consider an injured star graph with some faulty links and nodes. We show that even with $f_e \le n-3$ faulty links, a Hamiltonian cycle still can be found in an n-star, and that with $f_v \le n-3$ faulty nodes, a ring containing at most $4f_v$ nodes less than that in a Hamiltonian cycle can be found (i.e., the ring contains at least $n! - 4f_v$ nodes). In general, in an n-star with f_e faulty links and f_v faulty nodes, where $f_e + f_v \le n-3$, our embedding is able to establish a ring containing at least $n! - 4f_v$ nodes.

Index Terms—Fault tolerance, graph embedding, Hamiltonian cycle, interconnection network, processor allocation, ring, star graph.

1 Introduction

Designing and implementing multicomputer networks with versatile topologies, such as the linear array, ring, mesh, tree, hypercube, etc., have become possible due to fast advance in hardware technologies. One new interconnection network that has attracted a lot of attention recently is the star graph [1], [2]. Part of the reason is its symmetric and recursive nature, and superior (lower) node degree and comparable diameter as opposed to the hypercubes. Large numbers of references can be found in studying the star graph's topological properties [9], [22], embedding capability [12], [20], communication capability [3], [8], [10], [18], [19], [21], [24], [29], and fault-tolerant capability [4], [11], [14].

One of the central issues in evaluating a network is to study the *graph embedding problem* [5], [6]. Given a *guest graph* G and a *host graph* H, the problem is to find a mapping from each node of G to one of H, and a mapping from each edge of G to one path in H. This problem has long been used to model the problem of arranging a parallel algorithm in a parallel architecture. It also has applications in modeling the simulation of one parallel architecture by another.

The graph embedding problem has been heavily studied for various host graphs (see [13], [16] for more references). With a star graph as the host graph, any ring of an even length ≥ 6 has been shown to be embeddable [12]. Results regarding embedding multidimensional meshes into a star graph can be found in [12], [22]. The embedding of Hamiltonian cycles and hypercubes is discussed in [20].

As one can see, none of the above results discusses embedding in an injured star graph which has some faulty components. In this paper, we consider the problem of embedding a ring into an injured star graph which has some faulty links (or edges) and nodes (or vertices). Rings are common guest graphs with many applications (see [16], [17] for examples). Fault tolerance is an important issue in a multicomputer network, especially when the network is large. If, in a star graph, some components fail, it is desirable that the injured components be isolated from the rest of the network, so that the embedding is still possible. The similar problem of fault-tolerant ring embedding in hypercubes has also been studied in [7], [15], [23], [27], [26], [25].

In this paper, we develop embedding algorithms that utilize the hierarchical structure of an n-star. The embeddings achieved in this paper are summarized as follows:

1) with $f_e \le n-3$ faulty links, the embedding of a Hamiltonian cycle,
2) with $f_v \le n-3$ faulty nodes, the embedding of a ring containing, at most, $4f_v$ nodes less than that of a Hamiltonian cycle (in a fault-free n-star), and
3) with f_e faulty links and f_v faulty nodes, where $f_e + f_v \le n-3$, the embedding of a ring containing, at most, $4f_v$ nodes less than that of a Hamiltonian cycle.

One may notice that the first and second results are special (extreme) cases of the last result, when $f_v = 0$ and $f_e = 0$, respectively. While this is true, it is worth identifying these two cases because they are provably optimal (to be discussed in Section 7). Also, in our development, the third result is, in fact, derived by combining the techniques used in deriving the former two.

The rest of this paper is organized as follows. Preliminaries are given in Section 2. In Section 3, we develop a new scheme for finding a Hamiltonian cycle in a fault-free star graph. Note that the embedding scheme presented in Section 3 is different from those proposed in [12], [20], which also show the existence of a Hamiltonian cycle in a star graph. Our embedding is then extended with fault-tolerant capability when only links and only nodes may fail in Section 4 and Section 5, respectively. The result to tolerate both

• *The authors are with the Department of Computer Science and Information Engineering, National Central University, Chung-Li, 32054, Taiwan. E-mail: {yctseng, sheujp}@csie.ncu.edu.tw*

Manuscript received 13 Oct. 1995.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 101333.

link and node failures is presented in Section 6. Finally, conclusions are drawn in Section 7.

2 PRELIMINARIES

An *n-dimensional star graph*, also referred to as *n-star* or S_n, is an undirected graph consisting of $n!$ nodes (vertices) and $(n-1)\,n!/2$ links (edges). Each node is uniquely assigned a label $x_1x_2 \cdots x_n$, which is the permutation of n distinct symbols $\{x_1, x_2, \ldots, x_n\}$. Two nodes are joined by an edge *along* dimension d if the label of one node can be obtained from the other by swapping the first symbol and the dth symbol, $2 \le d \le n$. Without loss of generality, throughout we let these n symbols be $\{1, 2, \ldots, n\}$. A four-dimensional star graph S_4 is shown in Fig. 1.

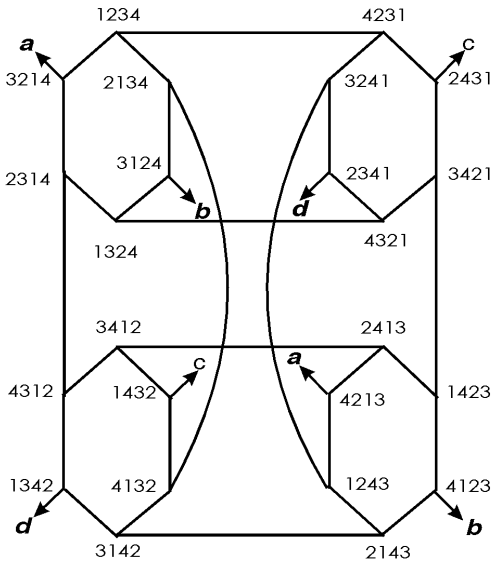

Fig. 1. A four-dimensional star graph S_4.

An S_n is a recursive structure that contains many smaller stars, or *substars*. Formally, a *k-dimensional substar*, or *k-substar*, $k \le n-1$, is denoted as a string $X = x_1x_2 \cdots x_n$, where $x_1 = *$ and $x_i \in \{*, 1, 2, \ldots, n\}$, $2 \le i \le n$. The symbol * means "don't care." In string X, there are exactly k *'s. Also, if $x_i \ne *$ and $x_j \ne *$, $1 \le i, j \le n$, then x_i and x_j must be distinct. The substar represented by X is a subgraph of S_n consisting of all legal ($k!$) vertices obtained from X by replacing each * with one digit in $\{1, 2, \ldots, n\}$, and edges induced by these vertices in S_n. For instance, **53* is a three-substar containing six nodes: 12534, 14532, 21534, 24531, 41532, and 42531.

DEFINITION 1. *Let $X = x_1x_2 \cdots x_j \cdots x_n$ be a k-substar with $x_j = *$. The j-cut on X, $j \ge 2$, is to partition X along the jth dimension into k copies of $(k-1)$-substars, each obtained from X by replacing x_j with a legal non-* symbol ("legal" in the sense that the symbol does not appear in the string X). Let $D = (d_1, d_2, \ldots, d_m)$, $m < k$, be a sequence of dimensions such that the $x_{d_i} = *$, $i = 1..m$. Then, the D-cut on X is to first apply a d_1-cut on X, whose result is then applied a $d_{\text{two-cut}}$, whose result is then applied a $d_{\text{three-cut}}$, etc. The final result is $k \times (k-1) \times \cdots \times (k-m+1)$ copies of $(k-m)$-substars.*

For instance, given a four-substar $X = ***5*3$ in an S_6, a three-cut on X is to partition X into four three-substars **15*3, **25*3, **45*3, and **65*3. If $D = (3, 5)$, a D-cut on X will apply a three-cut and then a five-cut on X. This generates the following two-substars: {**1523, **1543, **1563}, {**2513, **2543, **2563}, {**4513, **4523, **4563}, and {**6513, **6523, **6543}. Also note that in the above definition, if $j = 1$ then the partitioning result does not remain in substar structures.

DEFINITION 2. *Consider two k-substars X and Y in S_n. We define X and Y to be* adjacent *if their string representations differ in exactly one non-* position. If X and Y are adjacent, the* difference from *X to Y, denoted as dif(X, Y), is the symbol of X at the position where X and Y differ.*

Note that, by definition, two adjacent substars must have * symbols appearing in the same positions in their string representations. For instance, substar $X = **5*13*$ is adjacent to $Y = **5*23*$, but not adjacent to $Y' = **4*23*$. The difference from X to Y, or $dif(X, Y)$, is one, whereas $dif(Y, X)$ is equal to two.

The following discussion combines the notion of adjacency and cut, which reveals an essential technique used in this paper. Consider two adjacent k-substars $X = x_1 \cdots x_j \cdots x_n$ and $Y = y_1 \cdots y_j \cdots y_n$, such that $x_j = y_j = *$. If we apply a j-cut on X and Y, we will obtain k substars (of dimension $k-1$) from each of X and Y. By the above definition, one easily sees that all k substars (of dimension $k-1$) in X are adjacent to each other, since their string representations all differ in only one non-* position, and so are those k substars (of dimension $k-1$) in Y. Furthermore, among these substars, $k-1$ substars in X are adjacent to $k-1$ substars in Y in a one-to-one manner. Only the substar $x_1 \cdots x'_j \cdots x_n$ in X and the substar $y_1 \cdots y'_j \cdots y_n$ in Y are not adjacent, where $x'_j = dif(Y, X)$ and $y'_j = dif(X, Y)$ The idea is illustrated in Fig. 2, where the adjacency relation is represented by lines connecting substars. In particular, Fig. 2a shows three substars X, Y, and Z, with X adjacent to Y and Y adjacent to Z. Within each of X, Y, and Z, the three-substars are fully connected, while between X and Y (and similarly Y and Z) there are three connections. Also note that the three-substar ***256 in X, which is not connected to Y, satisfies $x'_j = 2 = dif(Y, X)$ (and, similarly, the three-substar ***526 in Y, which is not connected to X, satisfies $y'_j = 5 = dif(X, Y)$).

We summarize the above discussion with the following lemma.

LEMMA 1. *Consider any two adjacent k-substars X and Y and a legal j-cut on X and Y (by "legal", the jth symbol of X and Y must be *).*

a) *Suppose, after the cut, x_1, x_2, and x_3 are any three $(k-1)$-substars in X. Then, $dif(x_1, x_2) \ne dif(x_3, x_2)$.*
b) *Suppose, after the cut, x and y are any pair of adjacent $(k-1)$-substars in X and Y, respectively. Also, let x' be the only $(k-1)$-substar in X that is not ad-*

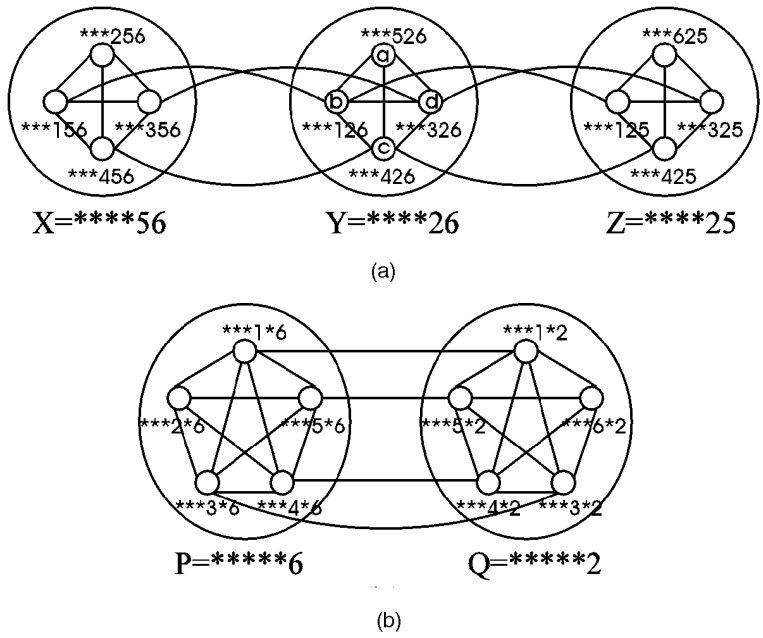

Fig. 2. The adjacency relation between (a) three adjacent four-substars X, Y, and Z, and (b) two adjacent five-substars P and Q. In both cases, a four-cut is applied.

jacent to any $(k-1)$*-substar in Y. Then, dif*(x, y) = *dif*(X, Y) *and dif*(x', x) = *dif*(Y, X) = *dif*(y, x).

DEFINITION 3. *A sequence of k-substars* $R = [X_0, X_1, \ldots, X_{r-1}]$ *is called a* k-ring *if substar* X_i *is adjacent to its neighbors* $X_{(i-1) \bmod r}$ *and* $X_{(i+1) \bmod r}$ *for any* $i = 0..r-1$.

In graph theory, a ring is simply a cycle of nodes. The above definition generalizes a ring with the notion of adjacency. For example, R = [***3*2, ***1*2, ***4*2, ***4*5, ***3*5] is a four-ring in an S_6. The following lemma, which will be heavily used in this paper, discusses how to obtain a $(k-1)$-ring from a k-ring.

LEMMA 2. *Given a k-ring* $R = [X_0, X_1, \ldots, X_{r-1}]$, $k \geq 4$, *it is possible to construct a* $(k-1)$*-ring* R' *of length kr from* R.

PROOF. First, we apply a legal j-cut on each X_i, $i = 0..r-1$, into k $(k-1)$-substars. We obtain kr substars. As mentioned earlier, in X_i, all substars are fully connected (in terms of adjacency) and there are $k-1$ connections between X_i and its neighbor $X_{(i+1) \bmod r}$. With so many connections, it is trivial to derive an R' which connects all kr substars by visiting substars in X_is along the direction of R. □

3 EMBEDDING OF A HAMILTONIAN CYCLE

In this section, we propose a new scheme for finding a Hamiltonian cycle in an S_n. Although the equivalent result has been established in [12], [20], our embedding is unique in that it utilizes the hierarchical structure of the star graph. Essentially, [12], [20] take a bottom-up approach, as follows. To construct a Hamiltonian cycle in an S_n, the existence of a Hamiltonian cycle in an S_{n-1} must be shown first. Then, the S_n is partitioned into n $(n-1)$-substars. The Hamiltonian cycle in S_n is formed by joining together the n Hamiltonian cycles in the n $(n-1)$-substars. On the contrary, in this paper, we take a top-down approach. Given an S_n, we first construct an $(n-1)$-ring, from which we will construct an $(n-2)$-ring, from which we will construct an $(n-3)$-ring, …, from which we will construct a one-ring, which is a Hamiltonian cycle. As will be seen in later sections, such an approach can be easily extended to a fault-tolerant embedding scheme.

Given an S_n, our embedding works as follows. First, we construct, from S_n, an $(n-1)$-ring. Then, we apply Lemma 2 to construct, from the $(n-1)$-ring, an $(n-2)$-ring. Then, we apply Lemma 2 to construct, from the $(n-2)$-ring an $(n-3)$-ring. This will be repeated recursively until a three-ring is obtained (however, note that, in the later stage of the construction, we will use embedding techniques stronger than what is used in Lemma 2 to generate rings with some special properties; this will be clear later). In the end, we will construct from the three-ring a one-ring, which is a Hamiltonian cycle.

In the following presentation, we will discuss the embedding backward from the last step (note that this is not in contradiction to our "top-down" approach).

3.1 Constructing a One-Ring from a Three-Ring

We first show how to construct a one-ring from a three-ring. Observe that there are two links between any two adjacent three-substars. These connections have two properties.

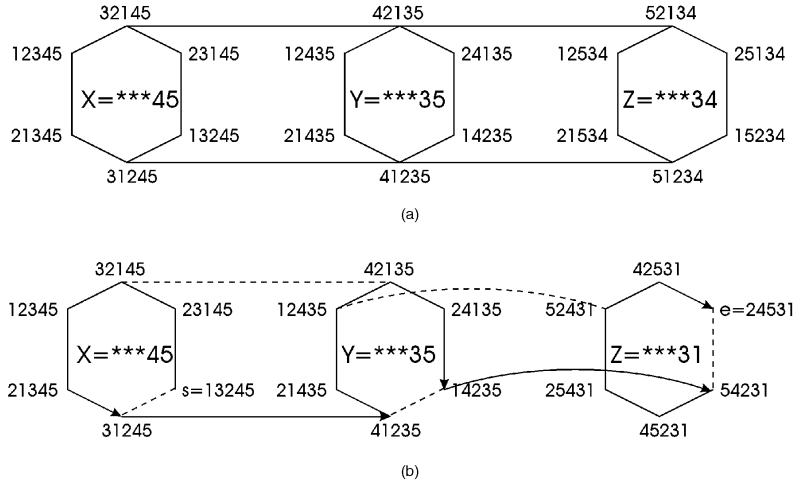

Fig. 3 . Three adjacent three-substars *X, Y, Z* in an S_5. In (a), condition *dif*(*X*, *Y*) = *dif*(*Z*, *Y*) holds and the graph is not Hamiltonian. In (b), *dif*(*X*, *Y*) ≠ *dif*(*Z*, *Y*) and a Hamiltonian path starting from node *s* to *e* can be found (shown in solid arrows).

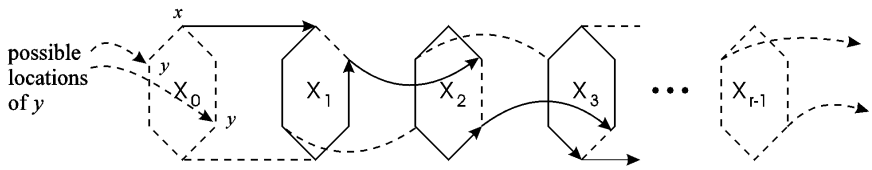

Fig. 4. Proof of Lemma 3.

P1: For any two adjacent three-substars *X* and *Y*, the two nodes in *X* connecting to *Y* are located at antipodal positions of the hexagon formed by *X* (i.e., the distance between these two nodes is three).

P2: Consider any three three-substars *X*, *Y*, and *Z* such that

(i) *X* is adjacent to *Y*,

(ii) *Y* is adjacent to *Z*, and

(iii) *dif*(*X*, *Y*) ≠ *dif*(*Z*, *Y*).

The two nodes in *Y* connecting to *X* are disjoint from those two in *Y* connecting to *Z*.

P2 can be proved as follows. The two nodes in *Y* connecting to *X* must have labels starting with symbol *dif*(*X*, *Y*). Similarly, the two nodes in *Y* connecting to *Z* have labels starting with *dif*(*Z*, *Y*), thereby proving the disjointness.

P2 (especially, condition (iii)) is important in finding a Hamiltonian cycle in our algorithm. To shed some light, Fig. 3a shows three adjacent substars *X*, *Y*, *Z* in an S_5 with *dif*(*X*, *Y*) = *dif*(*Z*, *Y*) = 4. The two nodes (42135 and 41235) in *Y*, having connections to *X*, must be the same as those having connections to *Z* (the proof is similar to the previous paragraph). It is easy to show that the graph formed by *X*, *Y*, *Z* does not contain a Hamiltonian path / cycle. On the contrary, in Fig. 3b, the condition *dif*(*X*, *Y*) ≠ *dif*(*Z*, *Y*) holds and the graph formed by *X*, *Y*, *Z* has a Hamiltonian path. In fact, by **P1** and **P2**, it is not hard to prove that, as long as the condition *dif*(*X*, *Y*) ≠ *dif*(*Z*, *Y*) holds, we can construct a path starting from *X*, visiting all nodes in *X*, connecting to *Y*, visiting all nodes in *Y*, connecting to *Z*, and then visiting all nodes in *Z*.

The above discussion leads to the following lemma.

LEMMA 3. *Given a three-ring* $R = [X_0, X_1, \ldots, X_{r-1}]$, *such that*

$$dif(X_{(i-1) \bmod r}, X_i) \neq dif(X_{(i+1) \bmod r}, X_i)$$

for any $i = 0..r-1$, *we can find a one-ring* R' *of length* $6r$ *from* R.

PROOF. We traverse the three-substars of *R* one after another. First, let *x* be any of the two nodes in X_0 that have a link connecting to X_1. We traverse, starting from *x*, visiting every node in X_1, and stopping at a node in X_1 with a link connecting to X_2 (see Fig. 4 for illustration). Properties **P1** and **P2** ensure that the above traversal is possible. We then visit every node in X_2 and stop at a node with a link to X_3. This process can be repeated until X_{r-1} is reached.

Suppose we stop at a node in X_{r-1} with a link connecting to a node, say *y*, in X_0. By **P1** and **P2**, the distance between *x* and *y* is either one or two (see Fig. 4). Now we need to traverse nodes in X_0. In the former case, a ring of length $6r$ can be easily formed. In the latter case, a ring of length $6r - 1$ will be formed, which is impossible because a star graph is bipartite [12] and any cycle must have an even length. Hence, the lemma. □

3.2 Constructing a 3-Ring from a 4-Ring

Earlier in Lemma 2, we have shown how to construct a three-ring from a given four-ring. However, care must be

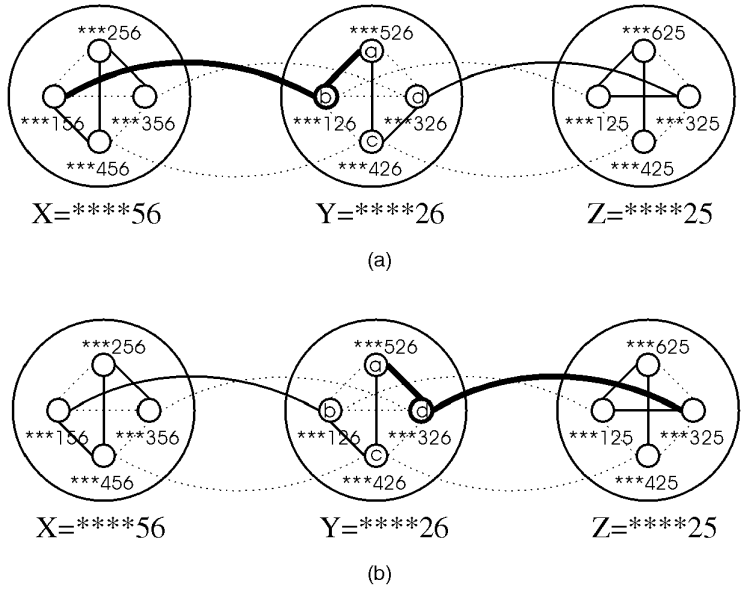

Fig. 5. The construction of a three-ring from a four-ring […, *X*, *Y*, *Z*, …], which satisfies *dif*(*X*, *Y*) = *dif*(*Z*, *Y*): (a) substar *a* is the second substar visited in *Y*, and (b) substar *a* is the third substar visited in *Y*. In both cases, the resulting three-rings violate property **P2** (the positions of violation are identified by bold lines).

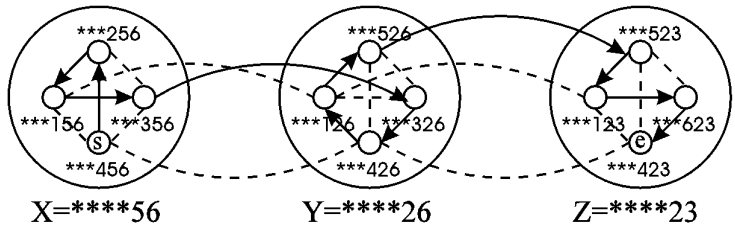

Fig. 6. Three adjacent four-substars *X*, *Y*, *Z* with *dif*(*X*, *Y*) = 5 ≠ *dif*(*Z*, *Y*) = 3. A path satisfying **P2** is shown by solid arrows from substar *s* to substar *e*.

taken to ensure that the three-ring satisfies **P2** so as to be used by Lemma 3. (By satisfying **P2**, we mean that every three consecutive three-substars in the ring have property (iii) in **P2**). For instance, suppose we have a four-ring […, *X*, *Y*, *Z*, …], such that *X*, *Y*, *Z* are as shown in Fig. 2a. Observe that the condition *dif*(*X*, *Y*) = *dif*(*Z*, *Y*) holds true. Suppose we want to construct a three-ring which traverses all three-substars in *X*, then all three-substars in *Y*, and then those in *Z*. The following reasoning shows that such a three-ring can not satisfy property **P2**.

1) As shown in Fig. 5a, substar *a* cannot be the first or last one visited in *Y*, since it has no connection to *X* and *Z*.
2) Because *dif*(*X*, *Y*) = 5, by Lemma 1b, the difference from any three-substar in *X* to any adjacent three-substar in *Y* is five. Further, by Lemma 1b, the difference from substar *a* to any of *b*, *c*, *d* is also *dif*(*X*, *Y*) = 5. So, *a* cannot be the second substar visited in *Y* (otherwise, **P2** is violated). The scenario is shown in Fig. 5a.
3) Similarly, because *dif*(*Z*, *Y*) is five, the difference from any three-substar in *Z* to any adjacent three-substar in *Y* is also five. Further, the difference from substar *a* to any of *b*, *c*, *d* is also *dif*(*Z*, *Y*) = 5. So, *a* can not be the third substar visited in *Y* (otherwise, **P2** is violated). The scenario is shown in Fig. 5b. This is a dilemma, since there is no proper position to put substar *a*.

As a counter-example, Fig. 6 shows three adjacent four-substars *X*, *Y*, *Z* with *dif*(*X*, *Y*) ≠ *dif*(*Z*, *Y*). A path satisfying **P2** can be found. This is formally reasoned below.

In general, consider any two adjacent four-substars *X* and *Y*. After applying an appropriate cut on *X* and *Y*, let *x* be the three-substar in *X* that does not have a connection to *Y*, and similarly, let *y* be the one in *Y* that does not have a connection to *X*. We propose two rules to visit the three-substars in *X* and *Y*:

R1: arrange *x* as the first or second substar traversed in *X*, and

R2: arrange *y* as the third or fourth substar traversed in *Y*.

These two rules are sufficient to ensure finding a three-ring satisfying **P2**. This is justified below (but it would be helpful for the reader to first verify these rules using the example in Fig. 6. To prove **R1**, first observe that any path in *X* must satisfy **P2**, even if we arbitrarily visit the substars in *X* (refer to Lemma 1a). By Lemma 1b, *dif*(*Y*, *X*) is the difference from any three-substar in *Y* to any three-substar in *X*. Suppose we arrange some x' (≠ *x*) and x'' (≠ *x*) as the third and fourth three-substars, respectively, visited in *X*. One

can easily show that *dif*(x', x'') must be not equal to *dif*(Y, X). So, a path established following rule **R1** must satisfy **P2**. We remark why x cannot be the third or last substar visited in X. Apparently, x cannot be placed as the last substar as there is no connection from x to Y. If x is placed as the third substar, then, by Lemma 1b, *dif*(x, x') = *dif*(Y, X) for any substar x' in X. As *dif*(Y, X) is the difference from any three-substar in Y to any three-substar in X, such a path will violate **P2**. This completes our remark.

A similar and symmetric argument can be extended to prove that a path in Y constructed by following **R2** will satisfy property **P2**.

It remains to identify the kind of four-rings that would enable us to apply rules **R1** and **R2** to construct a three-ring satisfying **P2**. This is formulated in the next lemma.

LEMMA 4. *Given a four-ring* $R = [X_0, X_1, \ldots, X_{r-1}]$, *such that*

$$dif(X_{(i-1) \bmod r}, X_i) \neq dif(X_{(i+1) \bmod r}, X_i)$$

for any $i = 0..r-1$, *it is possible to construct a three-ring* $R' = [X'_0, X'_1, \ldots, X'_{4r-1}]$ *from R such that*

$$dif(X'_{i-1 \bmod 4r}, X'_i) \neq dif(X'_{(i+1) \bmod 4r}, X'_i)$$

for any $i = 0..4r-1$.

PROOF. First, we apply any legal cut on R. Let x be any of the three three-substars in X_0 that have connections to X_1. We connect a path of three-substars from x to X_1, then to X_2, then to X_3, etc. In the process, rules **R1** and **R2** must be both followed.

Note that there is no conflict between **R1** and **R2** due to the condition $dif(X_{i-1}, X_i) \neq dif(X_{i+1}, X_i)$. Let x' and x'' be the three-substars in X_i that do not have a connection to X_{i-1} and X_{i+1}, respectively. By **R2,** x' must be placed as the third or last three-substar in X_i, and, by **R1,** x'' must be placed as the first or second three-substar. One question raised is: What if $x' = x''$? By Lemma 1b, for any three-substar y in X_i, other than x' and x'', the conditions $dif(x', y) = dif(X_{i-1}, X_i)$ and *dif*(x'', y) = *dif*(X_{i+1}, X_i) hold. As $dif(X_{i-1}, X_i) \neq dif(X_{i+1}, X_i)$, x' and x'' must be distinct and, hence, there is no conflict between **R1** and **R2**.

When the path is built up to X_{r-1}, some care is needed besides following rules **R1** and **R2**. Suppose, after running rules **R1** and **R2**, x' and x'' are the third and last three-substars visited in X_{r-1}. If x'' is adjacent to the starting three-substar x (in X_0), it would be impossible to join the other three unvisited three-substars in X_0 into R'. If so, we can reverse the order x' and x'' being visited. This does not cause any problem because x' also has a connection to X_0 (by **R1**, the three-substar in X_{r-1} that has no connection to X_0 has already been visited as the first or second three-substar).

Now, only the second and third three-substars visited in X_0 remain yet to be determined. The three-substar, if any, that has no connection to X_1 (resp., X_{r-1}) can be visited as the second (resp., third) one. So, the lemma is proved. □

3.3 Constructing a 4-Ring from a 5-Ring

The next job is to construct, from a given five-ring, a four-ring which satisfies the condition described in Lemma 4. The following lemma shows that any five-ring can offer such possibility.

LEMMA 5. *Given any five-ring* $R = [X_0, X_1, \ldots, X_{r-1}]$, *it is possible to construct a four-ring* $R' = [X'_0, X'_1, \ldots, X'_{5r-1}]$ *from R such that* $dif(X'_{(i-1) \bmod 5r}, X'_i) \neq dif(X'_{(i+1) \bmod 5r}, X'_i)$ *for any* $i = 0..5r-1$.

PROOF. First, we apply any legal cut on R. We will traverse the four-substars in X_0, X_1, …, X_{r-1} in that order. For any two adjacent five-substars X and Y in R, let x be the four-substar in X that does not have a connection to Y, and y the one in Y that does not have a connection to X. Similar to **R1** and **R2**, we use two rules to construct our four-ring:

R1′: x is the first, second, or third four-substar visited in X, and

R2′: y is the third, fourth, or fifth four-substar visited in Y.

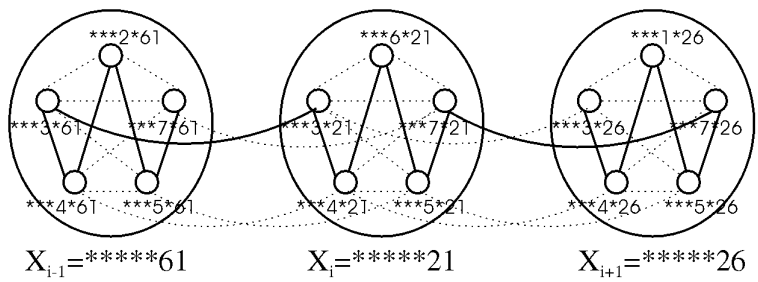

Fig. 7. The construction of a four-ring from a five-ring. Even if the condition $dif(X_{i-1}, X_i) = dif(X_{i+1}, X_i)$ holds, there is no conflict in satisfying both **R1′** and **R2′**, since we can still arrange the four-substar ***6*21 (which has no connection to both X_{i-1} and X_{i+1}) as the third substar traversed in X_i.

Using similar arguments for rules **R1** and **R2**, we can show that any four-ring constructed following rules **R1′** and **R2′** will satisfy our need. We omit the details. However, as opposed to Lemma 4, we note that this lemma does not rely on any relationship among X_{i-1}, X_i, X_{i+1} because even if the condition $dif(X_{i-1}, X_i) = dif(X_{i+1}, X_i)$ holds true, the four-substar in X_i that does not have a connection to both X_{i-1} and X_{i+1} still can be the third substar traversed in X_i. An example is in Fig. 7 to show such a scenario. The main reason is that there is an intersection position (the "third" position) in rules **R1′** and **R2′**, while no such intersection exists in rules **R1** and **R2**. □

3.4 The Embedding Algorithm

Below, we put together the above lemmas into a complete algorithm. The algorithm finds a Hamiltonian cycle in any S_n with $n \geq 6$.

Algorithm *Ham*();

1) Apply an n-cut on S_n. Construct an $(n-1)$-ring (referred to as R_{n-1}) of length n from S_n.
2) **for** $k = n-1$ **downto** 6 **do**
   Apply a k-cut on R_k and then use Lemma 2 to construct, from R_k, a $(k-1)$-ring (referred to as R_{k-1}).
3) Apply a five-cut on R_5 and construct, from R_5, a four-ring (referred to as R_4), using Lemma 5.
4) Apply a four-cut on R_4 and construct, from R_4, a three-ring (referred to as R_3), using Lemma 4.
5) Construct from R_3, a one-ring R_1, using Lemma 3.

Note that, when $n = 5$ (resp., 4), we can consider the S_5 (S_4) as a trivial five-ring R_5 (four-ring R_4) with a single node and directly run the algorithm from Step 3 (Step 4).

The embedding complexity can be analyzed as follows. Consider the construction of R_i, $i = n-1, n-2, \ldots, 3$. We need to break R_{i+1} into i-substars and traverse all of them. The traversal cost of each $(i+1)$-substar is proportional to the number of i-substars in this $(i+1)$-substar. So, the construction cost of R_i is proportional to the length of R_i. Similarly, the cost to construct R_1 is proportional to the length of R_1. This gives the embedding complexity of

$$n + n(n-1) + n(n-1)(n-2) + \cdots + \frac{n!}{3!} + n! = O(n!).$$

4 Ring Embedding when Some Links Fail

In the previous section, we have developed an algorithm *Ham*(), which can construct a Hamiltonian cycle in an S_n. The algorithm is not fault-tolerant, because any faulty component may destroy the cycle. In this section, we enhance *Ham*() to tolerate at least $f_e \leq n-3$ faulty links. We first show how to tolerate one faulty link in Lemma 3.

LEMMA 6. *In Lemma 3, if there exists a faulty link e which falls between two substars X_i and X_{i+1}, a one-ring R' can still be constructed without using link e.*

PROOF. Without loss of generality, we can assume that e falls between X_0 and X_1. Recall the proof of Lemma 3. The starting node x can be chosen as any node in X_0, with a link connecting to X_1. Simply choosing an x that is not incident by link e as the starting node would give an R' without passing e. □

Recall Lemma 2, Lemma 4, and Lemma 5, in each of which we discuss the construction of a $(k-1)$-ring from a k-ring for some k. Let's regard the connection between two adjacent $(k-1)$-substars as faulty if it contains at least one faulty link. Similarly, we can extend each of these lemmas with the capability of tolerating one faulty connection in the embedding.

LEMMA 7. *In Lemma 2, Lemma 4, and Lemma 5, if there exists a faulty edge e falling between two adjacent k-substars X_i and X_{i+1} in R, a $(k-1)$-ring R' still can be constructed without using link e.*

PROOF. Still, we assume without loss of generality, that e falls between X_0 and X_1. Recall the proofs of these lemmas. After an appropriate cut on R, link e will fall between some connection between a $(k-1)$-substar in X_0 and a $(k-1)$-substar in X_1. Simply starting our construction from a $(k-1)$-substar in X_0 that is not incident by this faulty connection will do the job. □

It is to be noted that, in the above lemma, e is not necessarily the only faulty link in R. However, avoiding e already serves our need in developing a fault-tolerant embedding.

Using Lemma 6 and Lemma 7, we can tolerate at least one faulty link in each construction from R_{n-1} to R_{n-2}, from R_{n-2} to R_{n-3}, …, from R_3 to R_1 (here, we are following the same notation as in *Ham*()). Thus, we should be able to tolerate at least $n-3$ faulty links.

However, to use these two lemmas, we need to make sure that the faulty links are falling between two k-substars in R_k (observe that, on the contrary, faulty links may be "encapsulated" within some k-substars). This can be done by applying an appropriate cut on R_{k+1}. For instance, if a faulty link e along dimension j falls inside a $(k+1)$-substar in R_{k+1}, then we can apply a j-cut on R_{k+1} when constructing R_k. Then, two cases may happen:

1) e is not used in R_k at all (which is fine for us), or
2) e falls between two k-substars in R_k.

If it is the latter case, Lemma 6 and Lemma 7 already provide the possibility of avoiding e in the construction from R_k to R_{k-1} (note the change of indices in the process: a cut on R_{k+1} along the dimension of e should be made first, and then, later, e can be avoided in the construction from R_k to R_{k-1}).

We summarize the embedding as follows. The algorithm works for any S_n, $n \geq 6$, with $f_e \leq n-3$ faulty links.

Algorithm *Link-Failure*();

1) Sort the $n-1$ dimensions of the S_n according to the numbers of faulty links falling on them in the descending order. Let $D = (d_n, d_{n-1}, \ldots, d_4)$ be the sequence of the first $n-3$ dimensions after the sorting.
2) Execute Steps 1 to 4 of algorithm *Ham*(), but apply a d_k-cut while constructing an R_{k-1} from R_k. Use Lemma 7 to avoid at least one (if any) faulty edge falling between two k-substars.

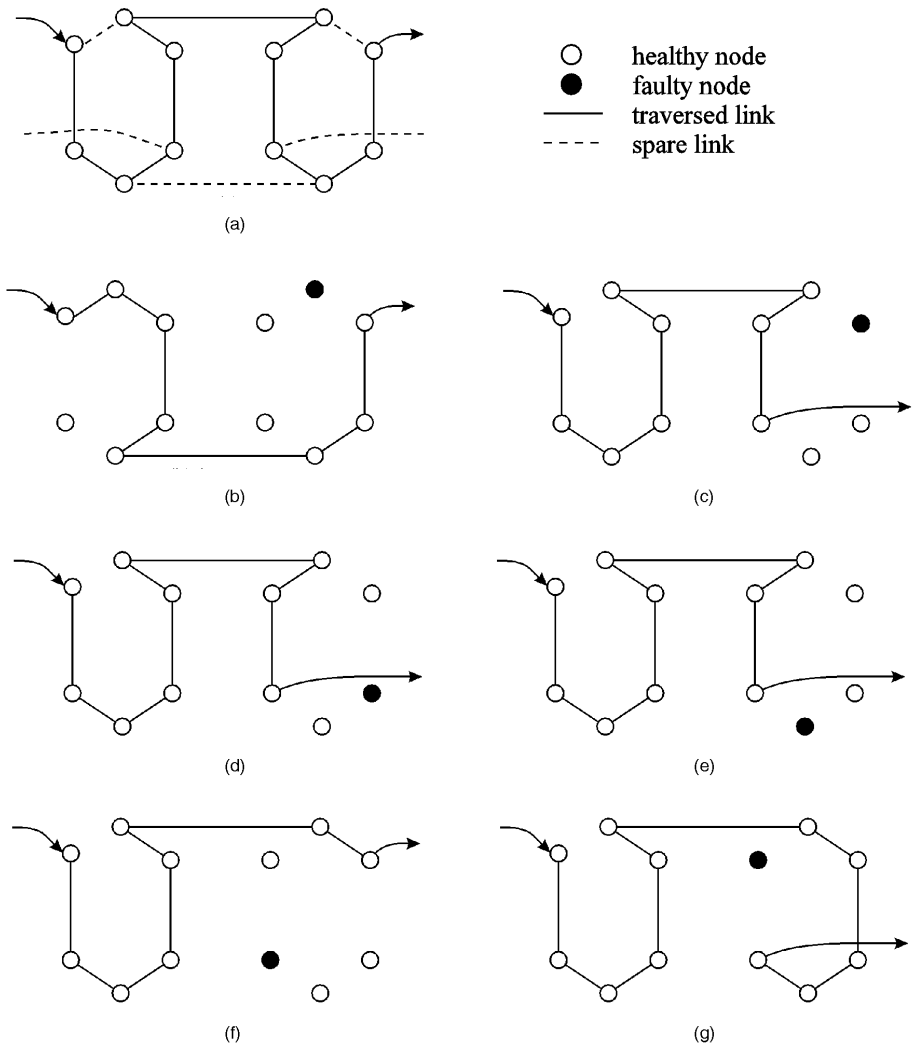

Fig. 8. (a) The original routing on two adjacent healthy three-substars, and (b)-(g) the fault-tolerant routing when one node in the second three-substar becomes faulty. (b) loss = 4, (c) loss = 3, (d) loss = 3, (e) loss = 3, (f) loss = 4, (g) loss = 1.

3) Construct from R_3 a fault-free one-ring R_1 using Lemma 6.

In Step 1, the number of faulty links along dimension d_i is no less than that along dimension d_{i-1}. The intention here is to let faulty links be cut (and, thus, be avoided) as early as possible. As each application of Lemma 6 and Lemma 7 is guaranteed to avoid only one link, without this sorting step, the resulting R_3 may contain more than one faulty link. Note that all faulty links will fall along dimensions d_n, d_{n-1}, …, d_4, as there are at most $n-3$ faulty links. Also, note that the above algorithm can be modified, as we have done for *Ham*() in Section 3, to run for cases of $n = 4$ and 5.

THEOREM 1. *Given an S_n, $n \geq 4$, with $f_e \leq n-3$ faulty links, algorithm Link-Failure() can find a fault-free Hamiltonian cycle in S_n.*

As to the embedding complexity, Step 1 will take $O(f_e \log f_e) = O(n \log n)$ for sorting. Steps 2 and 3 incur the same complexity, $O(n!)$, as by *Ham*(). This can be justified from the proofs of Lemma 6 and Lemma 7, where we only require that a faulty link fall between X_0 and X_1.

5 Ring Embedding when Some Nodes Fail

When there are some faulty nodes in an S_n, finding a Hamiltonian cycle is impossible. So, in this section, we study the following problem: Given an S_n with f_v faulty nodes, find a ring that is as large as possible without passing through any faulty node. Our main result shows that for any $f_v \leq n-3$ a ring of length at least $n! - 4f_v$ can be found.

We first consider the construction of a one-ring from a three-ring which has some faulty nodes. In Fig. 8a, we show two adjacent three-substars, through which a one-ring passes (indicated by solid lines). Now suppose one node in the second three-substar becomes faulty. In Fig. 8b-g, we show how to "route around" the faulty node under six possible fault scenarios. Note that the routing is based on a simple greedy strategy, by including as many nodes in these two three-substars as possible. As one can observe, the number of nodes (both faulty and nonfaulty) lost due to the failure is, at most, four. This leads to the following lemma.

LEMMA 8. *Given a three-ring* $R = [X_0, X_1, X_{r-1}]$ *in which*

a) *no two consecutive three-substars both contain faulty nodes,*

b) *each three-substar contains at most one faulty node, and*

c) $dif(X_{(i-1) \bmod r}, X_i) \neq dif(X_{(i+1) \bmod r}, X_i)$ *for any i,*

it is possible to construct a one-ring R' of length at least $6r - 4f$ *from R, where f is the total number of faulty nodes in R.*

PROOF. By a), let X_i be healthy and X_{i+1} contain a faulty node. By b), let x be any node in X_i whose neighbor in X_{i+1} is healthy. We traverse R from x toward substars X_{i+1}, …, X_{r-1}, X_0, …, X_{i-1}. In the traversal, when two consecutive three-substars are both healthy, we apply the technique used in Lemma 3 to connect all nodes. But, when there exists a faulty node, we should apply the greedy strategy as indicated in Fig. 8. As indicated earlier, we will lose, at most, four nodes per faulty node. So, we may lose, at most, $4f$ nodes.

When the path returns back to X_i, the end node may be at a distance of one or two from x (the scenario is similar to that in Fig. 4). As X_i is fault-free, in the former case, all nodes in X_i can be included in the ring, while, in the latter case, one more node will be excluded from the ring. Thus, the ring has a length $\geq 6r - 4f - 1$. As the ring length must be even, the lemma then follows. □

Note that, in the above lemma, condition a is essential (otherwise, one can easily construct a scenario in which finding a one-ring is prohibitive). In the next lemma, we point out when such a desired three-ring can be found from a four-ring.

LEMMA 9. *Given a four-ring* $R = [X_0, X_1, X_{r-1}]$ *in which*

1) *one of the four-substars is fault-free,*

2) *each four-substar contains at most one faulty node, and*

3) $dif(X_{(i-1) \bmod r}, X_i) \neq dif(X_{(i+1) \bmod r}, X_i)$ *for any i,*

it is possible to construct from R a three-ring R' satisfying the conditions (a)–(c) in Lemma 8.

PROOF. We apply the techniques used in Lemma 4 to construct an R' from R. Clearly, by conditions 2 and 3, R' will satisfy the conditions b and c in Lemma 8. It remains to ensure condition a, which can be done by interleaving faulty three-substars by healthy ones, as shown below.

Without loss of generality, let X_0 be healthy (by condition 1). If we can enforce that the last three-substar visited in each X_i, $i = 1..r-2$ be healthy, then every faulty three-substar in R' will be interleaved by at least one healthy three-substar. Observe that the third and fourth three-substars visited in X_i must both have connections to X_{i+1} (recall rule **R1**, which requires that the three-substar in X_i without connection to X_{i+1} be visited in the first or second position). As one of these two three-substars must be healthy, we are allowed to reorder the way that they are traversed if the fourth three-substar contains a faulty node.

Note that the above rule need not be followed when traversing the three-substars in X_{r-1}. This is because X_0 is fault-free, and, thus, the three-substar in X_{r-1} containing a fault (if any) has already been interleaved by healthy three-substar(s). However, we still have to ensure that the last three-substar visited in X_{r-1} be not adjacent to the starting three-substar in X_0 (refer to the proof of Lemma 4). □

To prepare a four-ring used in the above lemma, we need the next lemma, which shows how to cut S_n into four-substars, each containing at most one faulty node.

LEMMA 10. *In an* S_n, $n \geq 4$, *with* $f_v \leq n - 3$ *faulty nodes, there always exists a D-cut,* $|D| = n - 4$, *on* S_n *which results in four-substars each containing at most one faulty node.*

PROOF. Let F be the set of faulty nodes, $|F| \leq n - 3$. If $|F| \leq 1$, the lemma is trivially true. Otherwise, we can apply an appropriate j-cut on S_n so that F falls into at least two $(n-1)$-substar. (Such a j is easy to find. For instance, if F = {123456, 123654}, a four-cut or six-cut will work. In general, we simply select a j such that, in F, there exist two nodes whose jth symbols differ.)

After the above j-cut, let F be split into m nonempty subsets F_1, F_2, …, F_m, each falling in one $(n-1)$-substar. If each subset has a cardinality of one, then we are done. Otherwise, we can select any F_i such that $|F_i| \geq 2$, and apply another j'-cut so as to split F_i further into more subsets. This process can be repeated recursively. Clearly, $|D| = n - 4$ cuts are sufficient to partition $f_v \leq n - 3$ elements into subsets each containing a single element. □

EXAMPLE 1. Consider an S_7 with faulty set F = {1234567, 1342567, 4312567, 4321657}. We examine from position 7 to position 2. A seven-cut will not work because all faulty nodes will still fall in one six-substar. So, we apply a six-cut, which splits F into subsets F_1 = {1234567, 1342567, 4312567} and F_2 = {4321657}. Next, we need to split F_1. However, a five-cut will not work. So, we apply a four-cut, which splits F_1 into subsets F_{11} = {1234567} and F_{12} = {1342567, 4312567}. Finally, a three-cut can split F_{12} into two subsets. So, a D-cut with D = (6, 4, 3) is the desired cut.

Below, we summarize the above discussion into an algorithm for ring embedding in an S_n, $n \geq 6$, with $f_v \leq n - 3$ faulty nodes.

Algorithm *Node-Failure*();

1) Use Lemma 10 to find a sequence of dimensions $D = (d_n, d_{n-1}, \ldots, d_4)$.
2) Execute Steps 1 to 3 of algorithm *Ham*(), but apply a d_k-cut in the construction from R_k to R_{k-1}.
3) Construct, from R_4, a three-ring R_3, using Lemma 9.
4) Construct, from R_3, a 1-ring R_1, using Lemma 8.

After steps 1 to 3, a four-ring R_4 is obtained. Note how conditions 1-3 in Lemma 9 are satisfied. Condition 3 is

guaranteed by algorithm *Ham*(). Condition 2 is ensured by Lemma 10. Condition 1 holds because the number of four-substars $(n-4)! > n-3 \geq f_v$ for any $n \geq 6$. The correctness then follows directly from Lemma 9 and Lemma 8. Again, note that the above algorithm can be easily modified to run for cases of $n = 4$ and 5.

THEOREM 2. *Given an S_n, $n \geq 4$, with $f_v \leq n-3$ faulty nodes, Algorithm Node-Failure() can find a fault-free ring of length $\geq n! - 4f_v$.*

As to the embedding complexity, Step 1 takes time $O(f_v^2) = O(n^2)$. The cost of Step 2 is the same as that of *Ham*(). The cost of Steps 3 and 4 is also the same as that in *Ham*(), because, in the proofs of Lemma 9 and Lemma 8, the construction only involves local adjustment on the traversal of adjacent substars (four-substars and three-substars). So the total cost is bounded by $O(n!)$.

6 Ring Embedding When Both Links and Nodes Fail

In the previous two sections, we assume that either links or nodes may fail. In this section, we extend our result to deal with both cases simultaneously. We consider the problem of embedding a large ring in an S_n with $f_e \geq 1$ faulty links and $f_v \geq 1$ faulty nodes, where $f_e + f_v \leq n-3$. Without loss of generality, throughout we assume that $f_e + f_v = n-3$ (otherwise, we can arbitrarily regard some healthy links as faulty to satisfy this condition).

The algorithm combines the techniques used in *Link-Failure*() and *Node-Failure*(). Mainly, we will first apply a D_e-cut, which is then followed by a D_v-cut, on S_n, to obtain a four-ring. The lengths of sequences D_e and D_v are f_e and $f_v - 1$, respectively. To tolerate the f_e faulty links, we will generate a sequence of rings $R_{n-1}, R_{n-2}, \ldots, R_{n-f_e}$ using a D_e-cut. These f_e cuts make faulty links fall between substars in these rings. Using the techniques in *Link-Failure*(), at least one faulty link will be avoided in each construction from R_k to R_{k-1}, $k = n-1..n-f_e$, unless the ring is fault-free. Note that this may leave, at most, one faulty link in the last ring R_{n-f_e}; the faulty link (if any) must fall between two adjacent $(n-f_e)$-substars in R_{n-f_e} and will be avoided when constructing R_{n-f_e-1}.

To tolerate the f_v faulty nodes, we will generate, from the previous ring R_{n-f_e}, a sequence of rings $R_{n-f_e-1}, R_{n-f_e-2}, \ldots, R_4$ using a D_v-cut. These $f_v - 1$ cuts are sufficient to spread the f_v faulty nodes each into a distinct four-substar. The resulting R_4 will satisfy Lemma 9. Finally, using the techniques in *Node-Failure*(), a ring of length $\geq n! - 4f_v$ can be found. Note that, as Algorithm *Node-Failure*() is not able to tolerate any faulty link, we make a slight adjustment on it as follows: In the construction from R_{n-f_e} to R_{n-f_e-1}, if there exists one faulty link in R_{n-f_e}, then Lemma 7 should be used to avoid this faulty link.

The embedding algorithm is outlined below.

Algorithm *Link-Node-Failure*();

1) Sort the $n-1$ dimensions of S_n according to the numbers of faulty links falling on them in the descending order. Let $D_e = \left(d_n, d_{n-1}, \ldots, d_{n-f_e+1}\right)$ be the sequence of the first f_e dimensions after the sorting. Run Algorithm *Link-Failure*() to obtain rings $R_{n-1}, R_{n-2}, \ldots, R_{n-f_e}$, but apply cuts as specified in D_e.
2) Consider the faulty nodes in the $(n-f_e)$-substars of R_{n-f_e}. Use Lemma 10 to find a D_v-cut on these $(n-f_e)$-substars to obtain four-substars each containing, at most, one faulty node. Run Algorithm *Node-Failure*() to obtain rings $R_{n-f_e-1}, R_{n-f_e-2}, \ldots, R_4$.
3) Run the last two steps of Algorithm *Node-Failure*() to generate R_3 and R_1.

THEOREM 3. *Given an S_n with $f_e \geq 1$ faulty links and $f_v \geq 1$ faulty nodes, where $f_e + f_v \leq n-3$, Algorithm Link-Node-Failure() can find a fault-free ring of length $\geq n! - 4f_v$.*

7 Conclusions

In this paper, we have shown how to find a ring in an injured n-star. With $f_e \leq n-3$ faulty links, we prove that a Hamiltonian cycle can always be embedded in an n-star. It is impossible to find a Hamiltonian cycle if there exist $n-2$ faulty links all incident to a same node. So, the degree of fault tolerance, $n-3$, provided in this paper is optimal. With $f_v \leq n-3$ faulty nodes, we are able to find a large ring that may sacrifice at most $4f_v$ nodes in the network. The number of nodes sacrificed is optimal, within a factor of two, as in a bipartite graph (to which star graphs belong) the number of nodes sacrificed in a ring is at least $2f_v$ in the worst case. We have also combined the above two results and shown that, with f_e faulty links and f_v faulty nodes, where $f_e + f_v \leq n-3$, a large ring that may sacrifice, at most $4f_v$ nodes can be found.

We review some related works on fault-tolerant ring embedding in a hypercube below. In [15], [23], [28], it is shown that, with $n-2$ faulty links, a Hamiltonian cycle still can be found in a binary n-cube. In [7], it is shown that with $f \leq \left\lfloor \frac{n+1}{2} \right\rfloor$ faulty nodes, a ring of length at least $2^n - 2f$ can be found. In [25], it is shown that, given a binary n-cube with $f_e \leq n-4$ faulty edges and $f_v \leq n-1$ faulty vertices, such that $f_e + f_v \leq n-1$, a ring of length at least $2^n - 2f_v$ can be obtained. However, the derivation in this paper for star graphs is more complicated than that in [25] for hypercubes. It is interesting to compare the similarity between the results cited above for the fault-tolerant ring embedding in hypercubes and the results established in this paper for star graphs.

Acknowledgments

Dr. Yu-Chee Tseng's research is supported by the National Science Council of the Republic of China under Grant # NSC86-2213-E-008-029 and Grant # NSC86-2213-E-216-021.

Dr. Jang-Ping Sheu's research is supported by the National Science Council of the Republic of China under Grant # NSC85-2213-E-008-030.

REFERENCES

[1] S.B. Akers, D. Harel, and B. Krishnamurthy, "The Star Graph: An Attractive Alternative to the *n*-Cube," *Proc. Int'l Conf. Parallel Processing*, pp. 393-400, 1987.

[2] S.B. Akers and B. Krishnamurthy, "Group-Theoretic Model for Symmetric Interconnection Networks," *IEEE Trans. Computers*, vol. 38, no. 4, pp. 555-565, Apr. 1989.

[3] S.G. Akl, K. Qiu, and I. Stojmenovic, "Fundamental Algorithms for the Star and Pancake Interconnection Networks with Applications to Computational Geometry," *Networks*, vol. 23, no. 4, pp. 215-225, July 1993.

[4] N. Bagherzadeh, N. Nassif, and S. Latifi, "A Routing and Broadcasting Scheme on Faulty Star Graphs," *IEEE Trans. Computers*, vol. 42, no. 11, pp. 1,398-1,403, Nov. 1993.

[5] F. Berman and L. Snyder, "On Mapping Parallel Algorithms into Parallel Architectures," *J. Parallel and Distributed Computing*, vol. 4, pp. 439-458, 1987.

[6] S. Bokhari, "On the Mapping Problem," *IEEE Trans. Computers*, vol. 30, pp. 207-214, 1981.

[7] M.Y. Chan and S.-J. Lee, "Distributed Fault-Tolerant Embeddings of Rings in Hypercubes," *J. Parallel and Distributed Computing*, vol. 11, pp. 63-71, 1991.

[8] T.-S. Chen, Y.-C. Tseng, and J.-P. Sheu, "Balanced Spanning Trees in Complete and Incomplete Star Graphs," *IEEE Trans. Parallel and Distributed Systems*, vol. 7, no. 7, pp. 717-723, July 1996.

[9] K. Day and A. Tripathi, "A Comparative Study of Topological Properties of Hypercubes and Star Graphs," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, no. 1, pp. 31-38, Jan. 1994.

[10] P. Fragopoulou and S.G. Akl, "Optimal Communication Algorithms on Star Graphs Using Spanning Tree Constructions," *J. Parallel and Distributed Computing*, vol. 24, pp. 55-71, 1995.

[11] Z. Jovanovic and J. Misic, "Fault Tolerance of the Star Graph Interconnection Network," *Information Processing Letters*, vol. 49, pp. 145-150, 1994.

[12] J.-S. Jwo, S. Lakshmivarahan, and S.K. Dhall, "Embeddings of Cycles and Grids in Star Graphs," *Proc. Symp. Parallel and Distributed Processing*, pp. 540-547, 1990.

[13] T.-H. Lai and W. White, "Mapping Pyramid Algorithms into Hypercubes," *J. Parallel and Distributed Computing*, vol. 9, no. 1, pp. 42-54, 1990.

[14] S. Latifi, "On the Fault-Diameter of the Star Graph," *Information Processing Letters*, vol. 46, pp. 143-150, 1993.

[15] S. Latifi, S. Zheng, and N. Bagherzadeh, "Optimal Ring Embedding in Hypercubes with Faulty Links," *Proc. Fault-Tolerant Computing Symp.*, pp. 178-184, 1992.

[16] F.T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays-Trees-Hypercubes*. Morgan Kaufmann, 1992.

[17] B.P. Lester, *The Art of Parallel Programming*. Prentice Hall, 1993.

[18] V.E. Mendia and D. Sarkar, "Optimal Broadcasting on the Star Graph," *IEEE Trans. Parallel and Distributed Systems*, vol. 3, no. 4, pp. 389-396, July 1992.

[19] J. Misic and Z. Jovanovic, "Communication Aspects of the Star Graph Interconnection Network," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, no. 7, pp. 678-687, July 1994.

[20] M. Nigam, S. Sahni, and B. Krishnamurthy, "Embedding Hamiltonians and Hypercubes in Star Interconnection Graphs," *Proc. Int'l Conf. Parallel Processing*, vol. III, pp. 340-343, 1990.

[21] K. Qiu, "Broadcasting on the Star and Pancake Interconnection Networks," *Proc. Int'l Parallel Processing Symp.*, pp. 660-665, 1995.

[22] K. Qiu, S.G. Akl, and H. Meijer, "On Some Properties and Algorithms for the Star and Pancake Interconnection Networks," *J. Parallel and Distributed Computing*, vol. 22, pp. 16-25, 1994.

[23] A. Sen, A. Sengupta, and S. Bandyopadhyay, "On Some Topological Properties of Hypercube, Incomplete Hypercube and Supercube," *Proc. Int'l Parallel Processing Symp.*, pp. 636-642, 1993.

[24] J.-P. Sheu, C.-T. Wu, and T.-S. Chen, "An Optimal Broadcasting Algorithm without Message Redundancy in Star Graphs," *IEEE Trans. Parallel and Distributed Systems*, vol. 6, no. 6, pp. 653-658, June 1995.

[25] Y.-C. Tseng, "Embedding a Ring in a Hypercube with Both Faulty Links and Faulty Nodes," *Information Processing Letters*, vol. 59, pp. 217-222, 1996.

[26] Y.-C. Tseng, "Resilient and Flexible Ring Embedding in an Injured Hypercube," *J. Information Science and Eng.*, vol. 12, no. 4, pp. 567-583, Dec. 1996.

[27] Y.-C. Tseng and T.-H. Lai, "Ring Embedding in an Injured Hypercube," *Proc. Int'l Conf. Parallel Processing*, vol. III, pp. 149-152, 1993.

[28] Y.-C. Tseng and T.-H. Lai, "On the Embedding of a Class of Regular Graphs in a Faulty Hypercube," *J. Parallel and Distributed Computing*, vol. 37, pp. 200-206, 1996.

[29] Y.-C. Tseng and J.-P. Sheu, "Toward Optimal Broadcast in a Star Graph Using Multiple Spanning Trees," *IEEE Trans. Computers*, vol. 46, no. 5, pp. 593-599, May 1997.

Yu-Chee Tseng received his BS and MS degrees in computer science from the National Taiwan University and the National Tsing-Hua University in 1985 and 1987, respectively. From 1989 to 1990, he worked for WANG Laboratory and D-LINK Inc. as a software engineer. He obtained his PhD in computer and information science from Ohio State University in January of 1994. From February of 1994 to July of 1996, he was with the Department of Computer Science, Chung-Hua Polytechnic Institute, Taiwan. Since August of 1996, he has been an associate professor in the Department of Computer Science and Information Engineering, National Central University. Dr. Tseng served on the program committee of the International Conference of Parallel and Distributed Systems, 1996. His research interests include parallel and distributed computing, fault-tolerant computing, parallel computer architecture, wireless network, and mobile computing.

Dr. Tseng is a member of the IEEE Computer Society and the ACM.

Shu-Hui Chang received her BS degree in computer science from Chung-Yuan University in 1989. She received her MS degree from the Department of Computer Science and Information Engineering, National Central University, Taiwan, in 1995. She is currently an engineer with DELTA Electronics, Inc. Her research interests include parallel and distributed computing and interconnection networks.

Jang-Ping Sheu received his BS degree in computer science from Tamkang University, Taiwan, Republic of China, in 1981, and his MS and PhD degrees in computer science from the National Tsing Hua University, Taiwan, Republic of China, in 1983 and 1987, respectively.

He joined the faculty of the Department of Electrical Engineering, National Central University, Taiwan, Republic of China, as an associate professor in 1987. He is currently a professor and head of the Department of Computer Science and Information Engineering, National Central University. From March to June of 1995, he was a visiting researcher at the IBM T.J. Watson Research Center, Yorktown Heights, New York. His current research interests include parallelizing compilers, interconnection networks, and mobile computing.

Dr. Sheu is a member of the IEEE Computer Society and Phi Tau Phi Society. He is an associate editor of the *Journal of Information Science and Engineering* and the *Journal of the Chinese Institute of Electrical Engineering*. He received Distinguished Research Awards of the National Science Council of the Republic of China in 1993-1994 and 1995-1996.