Computer Science
Technical Report

A New Fixed Degree Regular Network for Parallel Processing*

Shahram Latifi
Department of Electrical Engineering
University of Nevada
Las Vegas, NV 89154
latifi@ee.unlv.edu

Pradip K. Srimani
Department of Computer Science
Colorado State University
Ft. Collins, CO 80523
srimani@cs.colostate.edu

Technical Report CS-96-126

Computer Science Department
Colorado State University
Fort Collins, CO 80523-1873

Phone: (970) 491-5792 Fax: (970) 491-2466
WWW: http://www.cs.colostate.edu

*To appear in the *Proceedings of Eighth IEEE Symposium on Parallel and Distributed Processing*, New orleans, October 1996.

A New Fixed Degree Regular Network for Parallel Processing*

Shahram Latifi
Department of Electrical Engineering
University of Nevada
Las Vegas, NV 89154
latifi@ee.unlv.edu

Pradip K. Srimani
Department of Computer Science
Colorado State University
Ft. Collins, CO 80523
srimani@cs.colostate.edu

Abstract

We propose a family of regular Cayley network graphs of degree three based on permutation groups for design of massively parallel systems. These graphs are shown to be based on the shuffle exchange operations, to have logarithmic diameter in the number of vertices, and to be maximally fault tolerant. We investigate different algebraic properties of these networks (including fault tolerance) and propose a simple routing algorithm. These graphs are shown to be able to efficiently simulate other permutation group based graphs; thus they seem to be very attractive for VLSI implementation and for applications requiring bounded number of I/O ports as well as to run existing applications for other permutation group based architectures.

1 Introduction

Performance of any distributed system is significantly determined by the choice of the underlying network topology. One of the most efficient interconnection networks has been the well known binary n-cubes or hypercubes; they have been used to design various commercial multiprocessor machines and they have been extensively studied. For the past several years, there has been a spurt of research on a class of graphs called Cayley graphs which are very suitable for designing interconnection networks. These Cayley graphs are based on permutation groups and they include a large number of families of graphs like star graphs [1, 2], hypercubes [3], pancake graphs [1, 4] and others [5]. These graphs are symmetric (edges are bidirectional), regular and seem to share many of the desirable properties like low diameter, low degree, high fault tolerance etc. with the well known hypercubes (which are also Cayley graphs).

All of these Cayley graphs are regular, i.e., each vertex has the same degree, but *the degree of the vertices increases with the size of the graph* (the number of vertices) either logarithmically or sub logarithmically. This property can make the use of these graphs prohibitive for networks with large number of vertices [6]. *Fixed vertex-degree* networks are also very important from VLSI implementation point of view [7]; there are applications where the computing vertices in the interconnection network can have only a fixed number of I/O ports [6]. As a matter of fact a large multiprocessor (with 8096 vertices) is being built by JPL around the binary De Bruijn graphs [8], which are *almost regular* network graphs of fixed vertex degree 4. There are also other graphs in the literature that have almost constant vertex degrees like the Moebius graphs [9] of degree 3. But neither the De Bruijn graphs [8] nor the Moebius graphs [9] are regular (and none are Cayley graphs). Also, these graphs have a low vertex connectivity of only 2 although most of the vertices in those graphs have degree larger than 2. Some fixed degree regular networks have been studied in [10, 11, 12].

In this paper, we introduce a Cayley graph for interconnecting a large number of processors but with a constant degree of the vertex. The proposed family of graphs is regular of degree 3 independent of the size of the graph, has a logarithmic diameter in the number of vertices and has a vertex connectivity of 3, i.e., the graphs are *maximally fault tolerant* (vertex connectivity cannot exceed the vertex degree). The proposed family of graphs offers a better and more attractive alternative to De Bruijn graphs or Moebius graphs for VLSI implementation in terms of regularity and greater fault tolerance. Another interesting feature of the proposed graphs is that it can easily simulate other permutation group based graphs and hence parallel algorithms designed for those topologies can be efficiently run on the new architecture with very minimal change.

*To appear in the ***Proceedings of Eighth IEEE Symposium on Parallel and Distributed Processing***, New orleans, October 1996.

2 Shuffle-Exchange Permutation (SEP) Graph

A Cayley graph is a vertex-symmetric graph often with $n!$ vertices, each assigned a label which is a distinct permutation of the set $\{1, 2, \ldots, n\}$. The adjacency among the vertices is defined based on a set of permutations referred to as *generators*; the neighbors of a vertex are obtained by composing the generators with the label of the vertex. The set of generators defined for a Cayley graph must be closed under the inverse operation to guarantee that the graph will be undirected. Furthermore, the set of generators together with the Identity permutation must produce all $n!$ permutations on n integers (which collectively belong to set S_n), or a subgroup of S_n when the generators are repeatedly applied to the already generated vertices.

An n-dimensional SEP graph is an undirected regular graph with $N = n!$ vertices, each vertex corresponding to a distinct permutation of the set $\{1, 2, \cdots, n\}$. Two vertices are directly connected if and only if the label (permutation) of one is obtained from the label of the other by one of the following operations:

- Swapping the first two digits (the leftmost digit is ranked as first).
- Shifting cyclically to left (or right) by one digit.

Formally, consider a set of n distinct symbols (we use English numerals as symbols; for example, when $n = 4$, the symbols are $1, 2, 3, 4$). Thus, for n distinct symbols, there are exactly $n!$ different permutation of the symbols or the number of vertices in SEP_n is $n!$; using $a_1a_2 \cdots a_n$ as the symbolic representation of an arbitrary vertex, the edges of SEP_n are defined by the following three generators in the graph:

$$g_L(a_1a_2 \cdots a_n) = a_2a_3 \cdots a_na_n$$
$$g_R(a_1a_2 \cdots a_n) = a_na_1a_2 \cdots a_{n-1}$$
$$g_{12}(a_1a_2 \cdots a_n) = a_2a_1a_3 \cdots a_n$$

Remark 1 *(i) The SEP is an undirected graph as swapping the first two digits is a symmetric operation; and if a vertex u is obtained from another vertex v by shifting to left, vertex v can be obtained from u by shifting to right; thus, if u is adjacent to v, so is v to u, (ii) The generator g_{12} is its own inverse, i.e., $g_{12}^{-1} = g_{12}$ and $g_L^{-1} = g_R$ ($g_R^{-1} = g_L$), (ii) Figure 1 shows the proposed trivalent Cayley graphs SEP_3 and SEP_4 of dimensions 3 and 4 respectively.*

Remark 2 *(i) The generator g_{12} performs the transposition (12) on the permutations and (ii) either g_L or g_R performs the n-cycle $(1, 2, \cdots, n)$ on the permutations.*

Lemma 1 *The transposition (12) and the n-cycle $(1, 2, \ldots, n)$ together generate S_n, the group of all permutations of n distinct elements.*

Proof : The proof follows from the fact that any basic 2-cycle of a permutation can always be written in terms of the swap and the n-cycle; for details, see [13]. □

Lemma 2 *SEP_n, $n \geq 2$, is a Cayley graph.*

Proof : The set of generators for SEP_n, $n \geq 2$, $G = \{g_{12}, g_L, g_R\}$ is closed under the inverse operation and the generator set can generate all the elements in the vertex set of SEP_n. □

Remark 3 *Note that the two generators $g_{(1,2)}$ and g_L could produce all the elements of the group S_n and so could the two generators $g_{(1,2)}$ and g_R. The third generator is included with two objectives: to make the generator set closed under inversion (a requirement for the graph to be a Cayley graph) and to make the graph undirected (bidirectional) since the bandwidth of a link incident on a vertex can be shared between the incoming traffic activated by g_L (or g_R) and the outgoing traffic activated by g_R (or g_L).*

Theorem 1 *For any n, $n \geq 2$, the graph SEP_n: (1) is a symmetric (undirected) regular graph of degree 3; (2) has $n!$ vertices; and (3) has $3n!/2$ edges.*

Proof : (1) follows from Remark 1. (2) follows from Lemma 1. (3) follows from (1) and (2). □

Definition 1 *The edges which correspond to g_{12} are called Exchange or E-edges and those corresponding to g_L (or g_R) are referred to as Shuffle or S-edges; an S-edge may be referred to either as an L-edge (corresponding to left shift) or an R-edge (corresponding to right shift).*

Remark 4 *This is similar to the familiar shuffle-exchange network where adjacent labels are obtained by either shifting the label cyclically to left by one digit, or by complementing the rightmost digit of the label (instead of swapping).*

Definition 2 *We use the regular expression $(E, L, R)^*$ to denote any sequence of generators; generator sequences will be used to specify vertex to vertex paths in the graph.*

3 Simple Routing Algorithm

Since SEP_n is a Cayley graph, it is vertex symmetric [1], i.e., we can always view the distance between any two arbitrary vertices as the distance between the source vertex and the identity permutation by suitably renaming the digits representing the permutations. Thus, in our subsequent discussion about a path from a source vertex to a destination vertex, the destination vertex is always assumed to be the identity vertex $I = (12 \cdots n)$ without any loss of generality. The

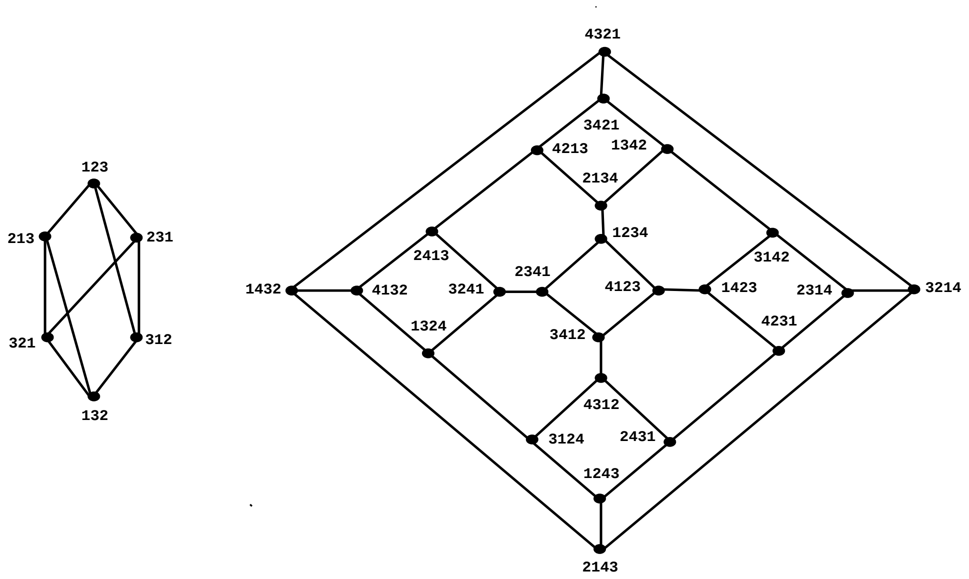

Figure 1. Example Graphs for $n = 3$ and $n = 4$

following algorithm Simple_Route computes a path from an arbitrary source vertex u to the identity vertex I.

The algorithm defines the moves to be taken at each step (either of the three generators). Each move takes us to a new vertex. The symbol set is given by $\{i|1 \le i \le n\}$ and for any digit i, we use $\rho(i)$ to denote the position of the digit i in the *current* vertex (the leftmost digit in a permutation has the position 1). For example, $\rho(5) = 3$ in the vertex (23541) while $\rho(1) = 5$ in the same vertex.

Algorithm Simple_Route

Step 1: If $\rho(1) > \lfloor n/2 \rfloor$, then apply g_R $(n - \rho(1))$ times; else apply g_L $\rho(1)$ times. [Digit "1" is now at the rightmost position.

Step 2: For i, $2 \le i \le \lfloor n/2 \rfloor$, do the following:

(A) If $n - \rho(i) + 1 < \rho(i) - 1$, then apply g_R $(n - \rho(i) + 1)$ times; apply the sequence of generators (g_{12}, g_L) $(n - \rho(i))$ times followed by a single g_L;

(B) otherwise, apply g_L $\max\{0, (\rho(i) - 2)\}$ times; if $\rho(i) > 1$, then apply g_{12}; apply the sequence of generators (g_R, g_{12}) $\max\{0, (\rho(i) - 2)\}$ times followed by a single g_L.

[Digits "1" through "i" are now at the rightmost positions of the current vertex]

Step 3: For digits i, $\lfloor n/2 \rfloor < i \le n - 1$, do the following: apply g_L $\max\{0, \max\{0, (\rho(i) - 2)\}$ times; if $\rho(i) > 1$, then apply g_{12}; apply the sequence of generators (g_R, g_{12}) $\max\{0, (\rho(i) - 2)\}$ times followed by a single g_L.

Step 4: Apply one g_L to reach the identity vertex.

Example: Consider the vertex (2143) in SEP_4. The path from this vertex to the identity vertex is computed by the algorithm as follows: $(2143) \xrightarrow{g_L} (1432) \xrightarrow{g_L} (4321) \xrightarrow{g_L} (3214) \xrightarrow{g_{12}} (2314) \xrightarrow{g_R} (4231) \xrightarrow{g_{12}} (2431) \xrightarrow{g_L} (4312) \xrightarrow{g_{12}} (3412) \xrightarrow{g_L} (4123) \xrightarrow{g_L} (1234)$. Thus the path length is 8.

Remark 5 *The algorithm is not optimal; note that the vertex node* (2143) *in* SEP_4 *can reach the identity vertex in 6 steps as follows:* $(2134) \xrightarrow{g_{12}} (1243) \xrightarrow{g_L} (2431) \xrightarrow{g_L} (4312) \xrightarrow{g_{12}} (3412) \xrightarrow{g_L} (4123) \xrightarrow{g_L} (1234)$. *Nevertheless it does establish a upper bound on the diameter of the graph, as we see below.*

We make the following observations about the worst case behavior of the above routing algorithm:

- Step 1 takes $\lfloor n/2 \rfloor$ moves.
- There are $\lfloor n/2 \rfloor - 1$ loops in Step 2. Consider an arbitrary value of i in the given range. Number of moves is maximum when $\rho(i) = \lfloor \frac{n+2}{2} \rfloor$ (using either subcase (A) or (B)). Number of moves for putting digit i in its correct place is $3(\rho(i) - 2) + 1 + 1 = 3\rho(i) - 4 = \frac{3n-2}{2}$. Note that for each value of i within the range specified in Step 2, this worst case

value of $\rho(i)$ is possible. Thus, maximum possible total number of moves in Step 2 is $\frac{1}{4}(n-2)(3n-2) = \frac{1}{4}(3n^2 - 8n + 4)$.

- Worst case in Step 3 occurs when $\rho(i) = n - i + 1$ for $\lfloor n/2 \rfloor < i \leq n-1$ and note that this possible. Thus, maximum possible total number of moves in Step 3 is

$$\sum_{j=n/2+1}^{n-1} \{1 + 3((n-j+1)-2) + 1\}$$

$$= \sum_{j=1}^{n/2-1} (3j-1) = \frac{1}{8}(3n^2 - 10n + 8)$$

- Step 4 always takes one move.
- In the worst case, the total number of moves made by the entire routing algorithm is given by (by summing up the moves from different steps) $\frac{1}{8}(9n^2 - 22n + 24)$.

Remark 6 *Obviously, the diameter of SEP_n for any given n, $n \geq 3$, is upper bounded by $\frac{1}{8}(9n^2 - 22n + 24)$. We strongly suspect that the actual diameter would be much less although most possibly $O(n^2)$.*

4 Algebraic Properties of SEP_n

In this section we investigate different interesting structural properties of the proposed 3-regular Cayley graphs.

Definition 3 *Any cycle in SEP_n consisting of only the s-edges (induced by the symmetric functions g_L or g_R) is called an s-cycle.*

Example: In Figure 1, the cycle $\{4312, 3124, 1243, 2431\}$ is an s-cycle.

Theorem 2 *All of the $n!$ vertices of SEP_n of dimension n are partitioned into vertex disjoint s-cycles of length n; number of s-cycles in SEP_n is $(n-1)!$.*

Proof : Consider an arbitrary vertex $v = a_1 a_2 \cdots a_n$ in SEP_n. For any i, $i \geq 1$, let $g_L^i(v) = g_L(g_L^{i-1}(v))$, where $g_L^1(v) = g_L(v)$. It is easy to observe that $g_L^n(v) = v$ and also $g_L^i(v) \neq g_L^j(v)$ for $1 \leq i, j \leq n$. Thus, from an arbitrary vertex v if the g_L (or the g_R) function is repeatedly applied, a cycle of length n is traced in the graph SEP_n.That these s-cycles are vertex disjoint follows from the fact that $g_L(v_1) = g_L(v_2)$, if and only if $v_1 = v_2$. □

Remark 7

- *Consider the symbol set $\{t_1, t_2, \cdots, t_n\}$ for SEP_n. For all k, $1 \leq k \leq n$, each s-cycle in SEP_n has a unique vertex starting with t_k, $1 \leq k \leq n$.*
- *For each s-cycle in SEP_n, the unique vertex starting with t_1 is called the* leader *vertex. Since there are n symbols and the leader nodes start with t_1, there are $(n-1)!$* leader *nodes in SEP_n which is equal to the number of s-cycles in SEP_n. For example, the leader node of the s-cycle cited in example 1 is 1243.*
- *Note that each leader node permutation starts with the digit "1". If we disregard the digit "1" in representing the leader nodes, each leader node ($(n-1)!$ of them) is labeled by a distinct permutation of $(n-1)$ distinct digits $\{2, 3, \cdots, n\}$.*

Definition 4 *Two s-cycles, say s_i and s_j, are said to be adjacent if there exists a vertex $v \in s_i$ and a vertex $u \in s_j$ such that $v = g_{12}(u)$ or $u = g_{12}(v)$.*

Theorem 3 *Each s-cycle in G_n is adjacent to n different s-cycles.*

Proof : Consider an arbitrary s-cycle with the leader $v = a_1 a_2 \cdots a_n$, where $a_1 = t_1$. Now, $g_{12}(v) = a_2 a_1 \cdots a_n = y_0$ and the node y_0 belongs to the f-cycle with leader $a_1 a_3 \cdots a_n a_2$. Then, consider the nodes y_i, $1 \leq i < n$, such that $y_i = g_{12}(v_i)$ where $v_i = g_L^i(v)$. We have $y_i = g_{12}(a_{i+1} a_{i+2} \cdots a_n a_1 a_2 \cdots a_i) = a_{i+2} a_{i+1} a_{i+3} \cdots a_n a_1 a_2 \cdots a_i)$; this node y_i belongs to a s-cycle with the leader $a_1 a_2 \cdots a_i a_{i+2} a_{i+1} a_{i+3} \cdots a_n$. Obviously, the nodes y_i, $0 \leq i < n$, belong to different s-cycles (they have different leaders) and hence any s-cycle is adjacent to n different s-cycles in SEP_n. □

Theorem 4 *For any vertex $v \in SEP_n$, $(LE)^{n-1}(v) = v$ and this defines a cycle of length $2(n-1)$ in SEP_n.*

Corollary 1 *For any vertex $v \in SEP_n$, $(RE)^{n-1}(v) = v$ and this defines a cycle of length $2(n-1)$ in SEP_n.*

Theorem 5 *For any vertex $v \in SEP_n$, $n > 3$, $(EREL)^3(v) = v$ and this defines a cycle of length 12 in SEP_n.*

5 Fault Tolerance of SEP_n

The node fault tolerance of an undirected graph is measured by the vertex connectivity of the graph. A graph G is said to have a vertex connectivity ξ if the graph G remains connected when an arbitrary set of less than ξ nodes are faulty. Obviously, the vertex connectivity of a graph G cannot exceed the minimum degree of a node in G; thus $\xi(SEP_n) \leq 3$ since SEP_n is a 3-regular graph for all values of n. A graph is called **maximally fault tolerant** if vertex connectivity of the graph equals the minimum degree of a node. Our purpose in this section is to show that the proposed graph SEP_n has a vertex connectivity of 3 and hence these graphs are maximally fault tolerant.

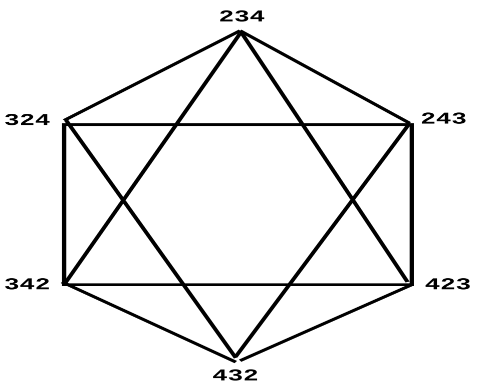

Figure 2. Reduced Graph RS_3 corresponding to SEP_4

Definition 5 *For a given SEP_n compute the reduced graph RS_{n-1} in the following way: condense each s-cycle into a single node and label that node with the leader node of the s-cycle without the digit "1" (see Remarks 7); connect two arbitrary vertices by an undirected edge iff the corresponding s-cycles are adjacent in SEP_n.*

Remark 8 *Figure 2 shows the reduced graph RS_3 corresponding to SEP_4. Each vertex in RS_{n-1} is labeled with a distinct permutation of $n-1$ distinct symbols; thus RS_{n-1} has $(n-1)!$ vertices each with a degree n (each vertex in RS_{n-1} corresponds to a distinct s-cycle in SEP_n); each s-cycle in SEP_n is of length n, has a leader node (digit "1" followed by a distinct permutation of $n-1$ digits, $\{2, 3, \cdots, n\}$), and is adjacent to n distinct other s-cycles.*

Lemma 3 *The reduced graph RS_{n-1} (corresponding to SEP_n) is a n regular graph with $(n-1)!$ nodes (each node representing a distinct permutation of $n-1$ digits $\{2, 3, \cdots, n\}$) with the generators g_L, g_R, and $E(i, i+1)$, for $1 \leq i < n$, where $E(i, i+1)$ indicates swapping of two adjacent digits i and $i+1$.*

Proof : SEP_n has $(n-1)!$ many s-cycles, each with a leader denoting a distinct permutation of $n-1$ digits $\{2, 3, \cdots, n\}$ and hence RS_{n-1} has $(n-1)!$ nodes. It remains to show the adjacency of the s-cycles is achieved by the said generators and nothing more.

- If two nodes u and v are adjacent in RS_{n-1}, they correspond to adjacent s-cycles in SEP_n. If $u = g_L(v)$, let $v = 23..n$ and $u = 34...n2$; restoring digit "1", we get the corresponding leader nodes $123...n$ and $134...n2$ (in SEP_n) of two s-cycles $s-1$ and s_2 say; s_2 has a node $2134...n$ which is connected to $123...n$ by an g_{12} edge in SEP_n and hence $s-1$ and s_2 are adjacent. Similar argument holds if $u = g_R(v)$. If $u = E(i, i+1)(v)$, let $v = 2...n$ and $u = 2..(i-1)(i+1)i(i+1)..n$. Again restoring digit "1", we get the corresponding leader nodes $12...n$ and $12..(i-1)(i+1)i(i+1)..n$ of two s-cycles $s-1$ and s_2 say; s_1 has a node $i(i+1)..n12..(i-1)$ which has an g_{12} edge in SEP_n to the node $(i+1)i(i+2)..n12..(i-1)$ that belongs to the s-cycle s_2; thus $s-1$ and s_2 are adjacent.
- If two s-cycles s_1 and s_2 are adjacent in SEP_n, their corresponding leader nodes are adjacent in RS_{n-1}; similar arguments, as before, hold in reverse direction.

□

Remark 9 *The reduced graph RS_{n-1} contains the bubble-sort graph B_{n-1} of dimension $n-1$ as a subgraph; a bubble-sort graph B_{n-1} of dimension $n-1$ is a $(n-2)$-regular Cayley graph with $(n-1)!$ vertices, each labeled with a distinct permutation of the set of integers $\{1, \ldots, n-1\}$ with a set of generators defined as: $G = \{g(i, i+1), i = 1, 2, \ldots (n-1)\}$ [1].*

Lemma 4 *Vertex connectivity of a bubble-sort graph B_{n-1} is $n-2$ [1].*

Lemma 5 *Vertex connectivity of RS_n, $n \geq 3$, is at least 3.*

Proof : For $n \geq 4$, RS_n contains B_n with connectivity $n-1$ (Lemma 4) and hence RS_n has vertex connectivity at least 3. For $n = 3$, exhaustive enumeration shows the result. □

Lemma 6 *Consider two arbitrary nodes u and v in SEP_n such that u and v belong to different s-cycles. Then there exist three vertex disjoint paths between u and v.*

Proof : Let $u \in s_i$ and $v \in s_j$ where $i \neq j$. Consider the following three s-cycles adjacent to s_i:

$$g_{12}(u) = u_1 \in s_{i1},\ g_{12}g_L(u) = u_3 \in s_{i2},\ g_{12}g_R(u) = u_4 \in s_{i3}$$

It is easy to see that for $n \geq 3$ these three s-cycles are distinct. Similarly, the node v can reach (by vertex disjoint paths) to three distinct s-cycles, say $s_{j1}, s_{j2},$ *and* s_{j3}. Note that it is possible for given i and j that $s_{i\ell} = s_{jk}$ for some ℓ and k. By Menger's theorem [14], given two sets of nodes V and U such that $|V| = |\{v_1, v_2, \cdots v_n\}| = |U| = |\{u_1, u_2, \cdots, u_n\}| = n$ in a n-connected graph, there are

n vertex disjoint paths connecting the nodes $(v_i \rightsquigarrow u_i)$, $1 \leq i \leq n$. The reduced graph RS_{n-1} corresponding to SEP_n is 3-connected by Lemma 5; hence for $n \geq 3$ there are 3 vertex disjoint paths connecting the two sets of 3 distinct s-cycles. Thus, there exist three vertex disjoint paths between u and v. □

Lemma 7 *Consider two arbitrary nodes u and v in SEP_n such that u and v belong to the same s-cycle. Then there exist three vertex disjoint paths between u and v.*

Proof : Since u and v belong to the same s-cycle s', we directly get two vertex disjoint paths between u and v along the given s-cycle s'. Consider the following two s-cycles:

$$g_{12}(u) = u_1 \in s_1, \; and \; g_{12}(v) = v_1 \in s_2$$

Note that s_1 and s_2 are distinct for $n \geq 3$ and neither of them is the same as s'. Now, by the 3-connectivity of the reduced graph RS_{n-1} corresponding to SEP_n, there exist 3 vertex disjoint paths between s_1 and s_2; at least two of them cannot go via the s-cycle s'. Hence, there exists a path between the nodes u_1 to v_1 in SEP_n that does not go through any of the nodes in the s-cycle u and v belong to. Thus there exist three vertex disjoint paths between u and v. □

Theorem 6 *The graph $SEP_n$is 3-connected for any given n, $n \geq 3$.*

Proof : Obvious from the previous two theorems. □

6 Network Simulations

In this section we consider the simulation of three Cayley networks namely, Bubble-sort, Pancake, and Star by SEP. Efficient network simulation strategies are necessary to run parallel algorithms on our proposed network SEP which were originally designed for those three networks [4]. We assume that the networks operate in the SIMD mode. At any given step of an algorithm, the controller broadcasts the instruction to be executed to all the processors. The processors then apply the same instruction to their own sets of data concurrently. In each case we determine the optimal number of steps required for one network to simulate the other. For network A to simulate network B, A should be able to perform each interconnection function offered by B in a finite number of steps. This simulation takes place by executing a sequence of interconnection functions provided by A. The simulation of B by A is denoted by $A \longrightarrow B$ (i.e., A simulates B). Programs, developed to run on network architecture B (i.e., that utilize the interconnection functions of B), can now run on network A by translating each interconnection function of B required by the program into an equivalent (but minimal) sequence of interconnection functions of A.

6.1 Star, Bubble-sort and Pancake

We choose three networks, e.g., Star, Bubble-sort and Pancake, since each of these networks is a Cayley graph with $n!$ vertices, each vertex labeled with a permutation of the first n non-zero positive integers. The significant difference between any of these networks and our proposed SEP network is that degree requirement of the nodes in each one of them increases with the size of the network while SEP is a fixed degree network for any size; this makes SEP much more attractive especially for VLSI implementations and for applications where the number of I/O ports per node is bounded and at the same time applications developed for other networks can run on SEP with minimal modification. Any Cayley graph is uniquely identified by the set of generators and the Identity element ($I = I = 12 \ldots n$).

We need three different kind of generators to define the networks under study: (1) transposition type generators g_{ij} which works on a permutation by swapping the ith and jth elements in the label; (2) shift type generators g_L or g_R which gives a cyclic shift to the elements in the label by one digit to left (right); (3) flipping type generators f_p ($2 \leq p \leq n$), that reverses the first p elements of the node permutation. For instance, the application of $g_{(2,5)}$ and f_3 to the permutation $I = 12345$ will yield permutations 15342 and 32145, respectively. Application of $g_L(g_R)$ to I will yield 23451 (51234). We now briefly define the three Cayley networks under study:

Star Graph S_n: $G_{star} = \{g_{(1,i)}; \quad 2 \leq i \leq n\}$, where $g_{(1,i)}$ is a generator that swaps the first and ith element.

Bubble-sort Graph B_n: $G_{B-sort} = \{g_{(i,i+1)}; \quad 1 \leq i < n\}$; this implies that a generator can only swap the adjacent digits in the label.

Pancake Graph P_n: $G_{Pancake} = \{f_i; \quad 2 \leq i \leq n\}$; a generator f_i in this graph flips the leftmost i digits.

SEP Graph SEP_n: $G_{SEP} = \{g_{(1,2)}, g_L, g_R\}$.

6.2 Basic Principle of Simulation

Note that there are two ways to specify a generator. One way is naturally to use the permutation obtained after the generator is applied to the Identity element I (absolute representation). The second way is to express this representation as a set of cycles (cycle representation). The latter uses the property that any permutation can be viewed as a set of cycles, i.e. cyclically ordered sets of digits with the property that each digit's desired position is that occupied by the next digit in the set. For instance, the abstract representation of g_L is: $(2, 3 \ldots n, 1)$ but its cycle representation is: $(1, 2 \ldots n)$. Similarly, the generator $g_{(i,j)}$ has the cycle representation of (i, j) which indicates the fact that in the permutation label the digits in positions i and j must be swapped. The cycle representation for other generators will be derived later. The decomposition of a cycle representation can be easily transformed to that of its corresponding generator. For example, $(i, j) = (1, i)(1, j)(1, i)$ implies $g_{(i,j)} = g_{(1,i)} - g_{(1,j)} - g_{(1,i)}$.

In the current context, the interconnection functions of a network are essentially its generators which are permutations themselves. Thus, the simulation problem can be reduced to one of expressing (or decomposing) the generators of B in terms of generators of A. The following assumptions are made:

- Each simulation involves movement of $N = n!$ data items among N processors.
- When simulating interconnection f, the data originally in processor p must be transferred to processor $f(p)$, $1 \leq p \leq N$.
- The simulation time ($t_{simulation}$) is in terms of the number of executions required to perform the simulation.
- The particular interconnection function to be simulated that require the most time to simulate will determine the simulation time.

Observe that no two (or more) processors can send data to the same processor when N data items are moved across any of the four networks in parallel and according to a single interconnection function. This is because every interconnection function is indeed a permutation. Applying this permutation to a distinct label of a given vertex can only yield a distinct label of another vertex. Formally, if $g_{(i,j)}P_1 = g_{(i,j)}P_2$ then $P_1 = P_2$, where P_1 and P_2 are permutations representing the labels of two vertices and $g_{(i,j)}$ is a generator representing an interconnection function. The potential of data conflict, however, exists when various interconnection functions are applied to different subnetworks.

6.3 SEP ⟶ Bubble-sort

The cycle $(k, k+1)$ can be decomposed as follows:

$$(k, k+1) = (1, 2, \ldots, n)^{k-1}(1, 2)(1, 2, \ldots, n)^{n-k+1}$$

The superscripts imply the repetition of the cycles. The generator $g_{(k,k+1)}$ can thus be expressed in one of the following ways:

$$\begin{aligned} g_{(k,k+1)} &= g_L^{k-1} - g_{(1,2)} - g_R^{k-1}; \qquad or \\ g_{(k,k+1)} &= g_R^{n-k+1} - g_{(1,2)} - g_L^{n-k+1}, \qquad 2 \leq k < n. \end{aligned}$$

The first simulation takes $(2k - 1)$ steps whereas the 2nd simulation takes $(2n - 2k + 3)$ steps. It follows that if $k \leq \lfloor n/2 \rfloor + 1$, the first approach should be used. Otherwise, the second approach will result in a faster simulation. The simulation time is $(n + 1)$ which is optimal. The optimality follows from the fact that in order to swap two adjacent digits, they must be first brought into 1st and 2nd positions where the swap is allowed (i.e. $g_{(1,2)}$). After the swap the digits must be put back in place by successive appropriate (left or right) shifts.

6.4 SEP ⟶ Pancake

We need to use an inductive approach. More specifically, assuming that the simulation of f_{i-1} is known, the steps to simulate f_i will be given. We use the simulation of f_3 as the induction base. The simulation of f_3 requires 5 steps and is given by:

$$f_3 = g_{(1,2)} - g_L - g_{(1,2)} - g_R - g_{(1,2)}$$

Now assuming the simulation of f_{i-1} is known, we will simulate f_i by first simulating f_{i-1} which will change the original label of $a_1 a_2 \ldots a_{i-1} a_i a_{i+1} \ldots a_n$ to $a_{i-1} a_{i-2} \ldots a_2 a_1 a_i a_{i+1} \ldots a_n$. We need only to bring a_i to the first position, and shift $a_{i-1}, a_{i-2}, \ldots, a_2$, and a_1 to the right by one position. It is left to the reader to verify that the following sequence will optimally make the change in the label as required.

$$g_L^{i-2} - g_{(1,2)} - (g_R - g_{(1,2)})^{i-2}$$

The total number of steps in the above sequence is: $3i - 5$. Thus, the simulation time can be obtained by solving a recurrence on $S(i)$, the number of steps to simulate f_i. We have $S(i+1) = S(i) + 3i - 5$ with $S(3) = 5$. Solution of this recurrence yields;

$$S(i) = \frac{3i^2 - 13i + 22}{2}, \quad i \geq 3$$

Coincidentally, from the above $S(2) = 1$ which is true since: $f_2 = g_{(1,2)}$. Since $S(n)$ will determine the maximum number of steps to simulate any pancake function, the simulation time is given by $t_{simulation} = \frac{3n^2 - 7n + 4}{2}$. The simulation time, in this case, is large, but optimal; This is probably due to the nature of specific generators of the Pancake graphs.

6.5 SEP ⟶ Star

We need to simulate the generators $g_{(1,i)}$, $2 \leq i \leq n$. To simulate $g_{(1,i)}$, the digit in the ith position is first brought to the first position such that the label $a_1 a_2 \ldots a_n$ be transformed to $a_1 a_i a_2 \ldots a_{i-2} a_{i-1} a_{i+1} \ldots a_n$. This can be done by applying the sequence $g_{(1,2)} - g_L$, $(i-2)$ times. Then the $g_{(1,2)}$ can be applied to obtain $a_i a_1 a_2 \ldots a_{i-2} a_{i-1} a_{i+1} \ldots a_n$. Applying the sequence of $g_R - g_{(1,2)}$ $(i-2)$ times will take a_1 to position i; i.e. the label $a_i a_2 \ldots a_{i-1} a_1 a_{i+1} \ldots a_n$ will be obtained. Thus,

$$g_{(1,i)} = (g_{(1,2)} - g_L)^{i-2} - g_{(1,2)} - (g_R - g_{(1,2)})^{i-2}$$

The simulation can also be done using the SEP generator g_R instead of g_L and then the sequence will look like

$$g_{(1,i)} = (g_R - g_{(1,2)})^{n-i+1} - (g_L - g_{(1,2)})^{n-i} - g_L$$

Depending on the location of i in the node label, one of the above schemes takes fewer steps, i.e., results in a faster simulation. Th e worst case occurs when $i = \frac{2n+5}{4}$. This simulation is optimal as no other sequence can simulate $g_{(1,i)}$ in fewer steps. Thus, the simulation time is computed at $i = \frac{2n+5}{4}$ and is given by $t_{simulation} = 2n - 2$.

7 Conclusion

We have proposed a new family of regular Cayley networks which is shown to be very competitive to build massively parallel systems. The proposed networks are maximally fault tolerant and hence are more resilient to node and link failures than the almost regular networks like DeBruijn graphs or Möbius graphs. Another interesting feature of the proposed family of networks is that it can efficiently simulate many of the popular permutation group based networks like star graphs, bubble-sort or pancake graphs. Thus, algorithms originally designed for those graphs can still run on the proposed networks and one can still get the advantage of the fixed degree of the network (independent of the size).

References

[1] S. B. Akers and B. Krishnamurthy. A group-theoretic model for symmetric interconnection networks. *IEEE Transactions on Computers*, 38(4):555–566, April 1989.

[2] S. B. Akers and B. Krishnamurthy. The star graph: an attractive alternative to n-cube. In *Proceedings of International Conference on Parallel Processing (ICPP-87)*, pages 393–400, St. Charles, Illinois, August 1987.

[3] L. Bhuyan and D. P. Agrawal. Generalized hypercube and hyperbus structure for a computer netwrk. *IEEE Transactions on Computers*, 33(3):323–333, March 1984.

[4] K. Qiu, S. G. Akl, and H. Meijer. On some properties and algorithms for the star and pancake interconnection networks. *Journal of Parallel and Distributed Computing*, 22(1), July 1994.

[5] K. Day and A. Tripathi. Arrangement graphs: a class of generalized star graphs. *Information Processing Letters*, 42:235–241, July 1992.

[6] C. Chen, D. P. Agrawal, and J. R. Burke. dBCube: a new class of hierarchical multiprocessor interconnection networks with area efficient layout. *IEEE Transactions on Parallel and Distributed Systems*, 4(12):1332–1344, December 1993.

[7] M. R. Samatham and D. K. Pradhan. The De Bruijn multiprocessor network: a versatile parallel processing and sorting network for VLSI. *IEEE Transactions on Computers*, 38(4):567–581, April 1989.

[8] D. K. Pradhan and S. M. Reddy. A fault tolerant communication architecture for distributed systems. *IEEE Transactions on Computers*, C-31(9):863–870, September 1982.

[9] W. E. Leland and M. H. Solomon. Dense trivalent graphs for processor interconnection. *IEEE Transactions on Computers*, 31(3):219–222, March 1982.

[10] S. B. Akers and B. Krishnamurthy. Group graphs as interconnection networks. In *Proceedings of FTCS-14*, pages 422–427, 1984.

[11] S. Latifi, M. M. Azevedo, and N. Bagherzadeh. The star connected cycles: a fixed degree network for parallel processing. In *Proceedings of the International Conference on Parallel Processing*, volume 1, pages 91–95, 1993.

[12] F. Preparata and J. Vuillemin. The cube-connected cycles: a versatile network for parallel computation. *Communications of ACM*, 24(5):30–39, May 1981.

[13] M. A. Armstrong. *Groups and Symmetry*. Springer-Verlag, 1988.

[14] F. Harary. *Graph Theory*. Addison-Wesley, Reading, MA, 1972.