Average Distance and Routing Algorithms in the Star-Connected Cycles Interconnection Network

Marcelo Moraes de Azevedo*, Nader Bagherzadeh, and Martin Dowd
Dept. of Electrical and Computer Engr. – University of California – Irvine, CA 92697-2625
Shahram Latifi
Dept. of Electrical and Computer Engr. – University of Nevada – Las Vegas, NV 89154-4026

Abstract

The star-connected cycles (SCC) graph was recently proposed as an attractive interconnection network for parallel processing, using a star graph to connect cycles of nodes. This paper presents an analytical solution for the problem of the average distance of the SCC graph. We divide the cost of a route in the SCC graph into three components, and show that one of such components is affected by the routing algorithm being used. Three routing algorithms for the SCC graph are presented, which respectively employ random, greedy and minimal routing rules. The computational complexities of the algorithms, and the average costs of the paths they produce, are compared. Finally, we discuss how the algorithms presented in this paper can be used in association with wormhole routing.

1. Introduction

An interconnection network is characterized by four distinct aspects: *topology*, *routing*, *flow control*, and *switching* [11]. The *topology* of a network defines how the nodes are interconnected by links, and is usually modeled by a graph. *Routing* determines the path selected by a packet to reach its destination, and is usually specified by means of a *routing algorithm*. *Flow control* deals with the allocation of links and buffers to a packet as it is routed through the network. *Switching* determines the mechanism by which data is moved from an incoming link to an outgoing link of a node (e.g., store-and-forward, circuit switching, virtual cut-through, and wormhole routing are examples of switching techniques found in parallel architectures).

In this paper, we continue the study of topological and routing aspects of the star-connected cycles (SCC) interconnection network [10], which was recently proposed as an attractive extension of the star graph [1]. An SCC graph is related to a star graph in the same way a cube-connected cycles graph [12] is related to a hypercube [13]. Namely, an SCC graph is formed from a star graph by replacing the nodes of the latter with cycles or rings of nodes. The SCC graph constitutes an efficient architecture for execution of parallel algorithms, which include broadcasting [2] and FFT [14]. Mesh algorithms are also supported in SCC graphs via embeddings [3]. The SCC graph inherits many of the interesting properties of the star graph [1], while employing at most three I/O ports per node. This last aspect categorizes the SCC graph as a *bounded-degree network* (other examples are in [12, 15]). Networks with bounded degree favor area-efficient VLSI layouts, and scale more easily than variable-degree networks.

Previously known topological aspects of SCC graphs include degree, symmetry, diameter, and fault-diameter, and were derived in [4, 10]. Here, we continue the study of these by investigating the *average distance* (or *average diameter*) of SCC graphs. Our interest in this property is twofold: 1) to obtain a metric for comparing the performance of routing algorithms, and 2) to provide continued characterization of the graph theoretical aspects of SCC networks.

In the absence of other network traffic, modern switching techniques (e.g., wormhole routing [6]) achieve a *communication latency* which is virtually independent of the selected path length [11]. In this ideal environment, the two factors which contribute to the communication latency experienced by a packet are the *start-up latency* and the *network latency* [11]. In a realistic environment in which congestion occurs, however, a third factor known as *blocking time* also contributes to the communication latency.

Regardless of the flow control and switching mechanisms being used in the network, congestion can usually be minimized if fewer links are used when routing a packet [5]. For communication-intensive parallel applications, the blocking time (and, consequently, the communication latency) is expected to grow with path length [5]. In such cases, a routing algorithm should ideally compute paths whose *average cost* matches the *average distance* of the network.

In this paper, we show that routes in an SCC graph may contain up to three classes of links, which we refer to as *lateral links*, *MI local links*, and *MB local links* (see Sec. 3 for definitions). Exact expressions for the average number

*This research was supported in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq - Brazil), under the grant No. 200392/92-1.

0-8186-7683-3/96 $05.00 © 1996 IEEE

of lateral links and *MI* local links between two nodes in an SCC graph, and an upper bound on the average number of *MB* local links, are derived. When combined, these expressions produce a tight upper bound on the average distance of the SCC graph.

We show that the number of *MB* local links is affected by the routing algorithm being used, and propose three different algorithms for the SCC graph: random, greedy, and minimal routing. The proposed routing algorithms are compared according to criteria such as *computational complexity* (which affects their implementation in hardware) and *average routing cost*, for which figures were obtained by means of simulation. The results obtained with the minimal routing algorithm provide exact numeric solutions for the average distance of SCC graphs. Our simulations indicate that the greedy routing algorithm performs close to the minimal routing algorithm, while requiring a smaller complexity. We show that the random routing algorithm presents the smallest complexity among the three algorithms described in this paper, and provide average and worst-case routing cost metrics for it. Finally, we discuss how the three algorithms can be implemented in combination with wormhole routing [6].

2. Background

2.1. The star graph

An n-dimensional star graph, denoted by S_n, contains $n!$ nodes which are labeled with the $n!$ possible permutations of n distinct symbols. In this paper, we use the integers $\{1, \ldots, n\}$ to label the nodes of S_n. A node $\pi = p_1 \ldots p_i \ldots p_n$ is connected to $(n-1)$ distinct nodes, respectively labeled with permutations $\pi_i = p_i \ldots p_{i-1} p_1 p_{i+1} \ldots p_n$, $2 \leq i \leq n$ (i.e., π_i is the permutation resulting from exchanging the symbols occupying the first and the i^{th} position in π) [1]. Each of these $(n-1)$ possible exchange operations is referred to as a *generator* of S_n. Two nodes π and π_i of S_n are connected by a link iff there is a generator g_i such that $\pi \cdot g_i = \pi_i$. The link connecting π and π_i is referred to as an i^{th}*-dimension link* and is labeled i. S_n has $(n-1) \cdot (n!/2)$ links. S_n is a regular graph with degree $\delta(S_n) = n - 1$ and diameter $\phi(S_n) = \lfloor 3(n-1)/2 \rfloor$. S_n is vertex- and edge-symmetric, and has hierarchical structure. The degree and diameter of S_n are sublogarithmic on the size of the graph [1], which makes the star graph compare favorably with the hypercube.

2.2. The star-connected cycles (SCC) graph

An n-dimensional SCC graph, denoted by SCC_n, is a bounded-degree variant of S_n [10]. SCC_n is formed by replacing each node of S_n with a *supernode*, i.e. a ring of $(n-1)$ nodes. The connections between nodes inside the same supernode are referred to as *local links*. Each supernode is connected to $(n-1)$ adjacent supernodes, using *lateral links* inherited from S_n. Figure 1 shows SCC_4.

Nodes in SCC_n are identified by a label $\langle i, \pi \rangle$, where i is an integer such that $2 \leq i \leq n$ and π is a permutation of n symbols. Two nodes $\langle i, \pi \rangle$ and $\langle i', \pi' \rangle$ are connected by a link $(\langle i, \pi \rangle, \langle i', \pi' \rangle)$ in SCC_n iff either: 1) $(\langle i, \pi \rangle, \langle i', \pi' \rangle)$ is a local link, i.e. $\pi = \pi'$ and $\min(|i-i'|, n-1-|i-i'|) = 1$, or 2) $(\langle i, \pi \rangle, \langle i', \pi' \rangle)$ is a lateral link, i.e. $i = i'$ and π differs from π' only in the first and the i^{th} symbols, such that $\pi(1) = \pi'(i)$ and $\pi(i) = \pi'(1)$.

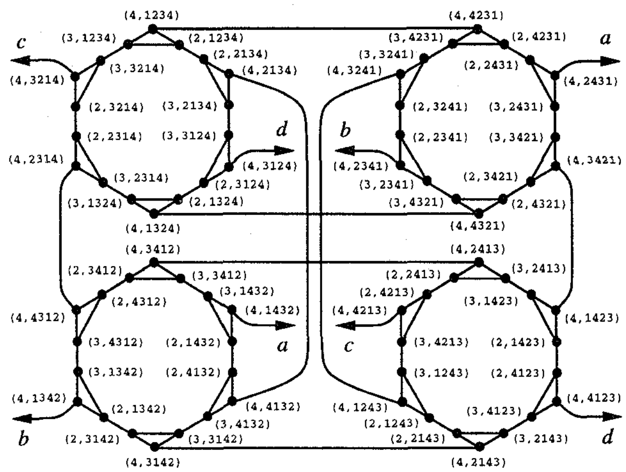

Figure 1. The SCC_4 graph

For similarity with S_n, the label of the supernode containing nodes $\langle 2, \pi \rangle, \ldots, \langle n, \pi \rangle$ is π. Also, the lateral link connected to node $\langle i, \pi \rangle$ is labeled i. For simplicity, supernode and lateral link labels are not shown in Fig. 1.

SCC_n contains $(n-1) \cdot n!$ nodes, $(n-1) \cdot n!$ local links, and $(n-1) \cdot (n!/2)$ lateral links. Thus, the size of SCC_n is comparable to that of S_{n+1}. Local links account for 2/3 of the links of SCC_n, and can be laid out very efficiently due to the ring topology of the supernodes. Moreover, SCC_n has about n times fewer lateral links than S_{n+1}, which further reduces the complexity of a VLSI layout for SCC_n when compared to S_{n+1}. SCC_n is vertex-symmetric, and has degree $\delta(SCC_n) = 2$ (for $n = 3$), and $\delta(SCC_n) = 3$ (for $n \geq 4$). In addition, the diameter of SCC_n is given by [10]:

$$\phi(SCC_n) = \begin{cases} 6, & \text{for } n = 3 \\ \frac{1}{2}(n^2 + n - 4), & \text{for even } n \\ \frac{1}{2}(n^2 + 3n - 8), & \text{for odd } n \geq 5 \end{cases} \tag{1}$$

3. Average distance of the SCC graph

3.1. Preliminaries

Let the cost of a route P between node $\langle i, \pi \rangle$ and the identity node $\langle i_0, \pi_0 \rangle = \langle 2, 12 \ldots n \rangle$ in SCC_n be $d = lat + loc$, where lat and loc respectively denote the number of lateral links and the number of local links in P. Because SCC_n is vertex-symmetric, its average distance can be computed by finding minimal cost routes to the identity from every node in the graph, and averaging those over $(n-1) \cdot n!$.

Before we can derive the average distance of SCC_n, some definitions related to lateral links are needed. We may organize the symbols of permutation π as a set of *r-cycles*[1] – i.e., cyclically ordered sets of symbols with the property that each symbol's desired position is that occupied by the next symbol in the set. In this paper, all r-cycles are written in canonical form [8] (i.e., the smallest symbol appears first in each r-cycle). For example, a permutation $\pi = 265431$ can be written in cyclic format as (1 2 6)(3 5)(4). Note that a symbol already in its correct position appears as a *1*-cycle.

Let $C_i = (i_0 \ \ldots \ i_{r-1})$ be an r-cycle in π, $2 \leq r \leq n$. Let $\pi \cdot R_i$ be the permutation produced from π by moving the symbols in C_i to their correct positions. The *execution of an r-cycle* C_i is, by definition, a minimal sequence of lateral links[2] R_i, leading from supernode π to supernode $\pi \cdot R_i$ (note that local links are not an issue here). R_i can be expressed by [7, 9]:

$$R_i = \begin{cases} (i_1, i_2, \ldots, i_{r-1}), & \text{if } i_0 = 1 \\ (i_0, i_1, \ldots, i_{r-1}, i_0), & \text{if } i_0 \neq 1 \end{cases} \tag{2}$$

In the case $i_0 \neq 1$, C_i can actually be executed with r different sequences of lateral links [7, 9]. Hence, for $j : 0 \rightarrow r-1$, such sequences can be expressed as:

$$(i_{j \bmod r}, i_{(j+1) \bmod r}, \ldots, i_{(j+r-1) \bmod r}, i_{j \bmod r}) \tag{3}$$

The minimum number of lateral links in a route from supernode π to π_0 does not depend on the order chosen to execute the r-cycles in π, and is given with [1]:

$$lat = \begin{cases} c + m, & \text{if } \pi\text{'s first symbol is 1} \\ c + m - 2, & \text{if } \pi\text{'s first symbol is not 1,} \end{cases} \tag{4}$$

where c is the number of r-cycles of length at least 2 in π and m is the total number of symbols in these r-cycles.

Routes in SCC_n often consist of sequences of lateral links interleaved with local links. In what follows, we give some definitions that relate to local links.

Recall that loc denotes the contribution of the local links to the total cost of a route P from $\langle i, \pi \rangle$ to $\langle i_0, \pi_0 \rangle$. loc can be further divided into two components, which we denote by $MI(loc)$ and $MB(loc)$, and define as follows:

- $MI(loc)$ – the number of *move-in (MI) local links* existing in the route from $\langle i, \pi \rangle$ to $\langle i_0, \pi_0 \rangle$. By definition, these are local links that must be traversed between two lateral links belonging to the execution sequence of an r-cycle in π.

- $MB(loc)$ – the number of *move-between (MB) local links* existing in the route from $\langle i, \pi \rangle$ to $\langle i_0, \pi_0 \rangle$. By definition, MB local links are: 1) local links that must be traversed between the executions of two consecutive r-cycles in π, 2) local links that must be traversed in supernode π, and are required to move from $\langle i, \pi \rangle$ to the lateral link that initiates the execution of the first r-cycle of π, and 3) local links that must be traversed in supernode π_0, and are required to move from the lateral link that finishes the execution of the last r-cycle of π to $\langle i_0, \pi_0 \rangle$.

Thus, $d = lat + loc = lat + MI(loc) + MB(loc)$. As an example, consider routing from $\langle 3, 34125 \rangle$ to $\langle 2, 12345 \rangle$ in SCC_5. The cyclic representation of permutation 34125 is (1 3)(2 4)(5). One possible route uses the sequences of lateral links $(2, 4, 2)$ and (3). Figure 2 shows the MI local links and the MB local links in such a route.

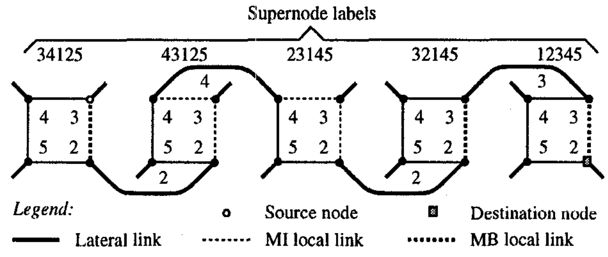

Figure 2. Types of links in a route in SCC_5

Note that from the topological viewpoint there is no distinction between MI and MB local links. A particular local link used by a route in SCC_n is considered to be either an MI or an MB local link, depending on the conditions stated above. Therefore, the same local link can be classified as an MI local link for some routes, and as an MB local link for others.

The cost components lat, $MI(loc)$, and $MB(loc)$ exist in any route in SCC_n (although in some short routes one or more of these components may be null). Due to vertex symmetry, one can derive the average distance of SCC_n by computing the average numbers of lateral links, MI local links, and MB local links in a route from $\langle i, \pi \rangle$ to $\langle i_0, \pi_0 \rangle$. We denote such average numbers by $\overline{lat}$, $\overline{MI(loc)}$, and $\overline{MB(loc)}$, respectively. The average distance of SCC_n, denoted by $\overline{\phi(SCC_n)}$, can then be expressed by:

$$\overline{\phi(SCC_n)} = \overline{lat} + \overline{MI(loc)} + \overline{MB(loc)} \tag{5}$$

Finally, the average number of local links existing in a route from $\langle i, \pi \rangle$ to $\langle i_0, \pi_0 \rangle$ in SCC_n is, by definition, $\overline{loc} = \overline{MI(loc)} + \overline{MB(loc)}$.

3.2. Average number of lateral links

The number of lateral links in the route between any node of SCC_n and the identity node is exactly equal to the cost of the corresponding route in the underlying n-star graph [10]. Therefore, $\overline{lat}$ is exactly equal to the average distance of S_n, which is given by [1]:

$$\overline{lat} = n + H_n + \frac{2}{n} - 4, \tag{6}$$

where $H_n = \sum_{k=1}^{n} \frac{1}{k}$ is the nth Harmonic number [8].

[1] *r-cycles* provide a convenient means to represent permutations [8] and should not be confused with *physical cycles or rings*, which constitute the supernodes of SCC_n.

[2] Throughout the paper, we distinguish the notation of an r-cycle from that of a sequence of lateral links by using commas in the latter.

3.3. Average number of *MI* local links

The number of *MI* local links in a route in SCC_n can be calculated as follows. Consider routing from $\langle i, \pi \rangle$ to the identity node $\langle i_0, \pi_0 \rangle$, and let the number of *r*-cycles of length at least 2 in π be c. Let $C_i = (i_0 \ \ldots \ i_{r-1})$ be one of these *r*-cycles, and let R_i be an execution sequence for C_i (Eq. 2). Moving between two consecutive lateral links i_a, i_b in R_i requires $d(i_a, i_b)$ *MI* local links, where [10]:

$$d(i_a, i_b) = \min\left(|i_a - i_b|, n - 1 - |i_a - i_b|\right) \tag{7}$$

The total number of *MI* local links that must be traversed during the execution of C_i, denoted by $MI(loc, C_i)$, is therefore the sum of the distances $d(i_a, i_b)$ between all pairs of consecutive lateral links (i_a, i_b) in R_i:

$$MI(loc, C_i) = \begin{cases} \sum_{j=2}^{r-1} d(i_{j-1}, i_j), & \text{if } i_0 = 1 \\ \sum_{j=1}^{r} d(i_{j-1}, i_{j \bmod r}), & \text{if } i_0 \neq 1 \end{cases} \tag{8}$$

Lemma 1 *The number of MI local links that must be traversed in a route between any two nodes of SCC_n is independent of the order chosen to execute the r-cycles existing between those nodes.*

Proof: We first show that $MI(loc, C_i)$ does not depend on the sequence of lateral links R_i chosen to execute C_i. If $i_0 = 1$, there is only one such sequence (Eq. 2). If $i_0 \neq 1$, there are r different possible sequences (Eq. 3). However, due to the cyclic nature of these sequences, they all have the same cost $MI(loc, C_i)$ (Eq. 8). By extension, the total number of *MI* local links in the route, $MI(loc)$, must also be an invariant. □

An immediate consequence of Lemma 1 is that the number of *MI* local links between two nodes of SCC_n can be derived without further considerations about routing. (Assuming, of course, that routing is accomplished in adherence to Eqs. 2 and 3, as is the case with all routing algorithms presented in this paper.) As an example, consider an *r*-cycle $C_i = (2\ 6\ 4)$, and let $n = 7$. C_i can be executed with a sequence of lateral links $R_i = (2, 6, 4, 2)$. The number of *MI* local links required in the execution of this sequence is $MI(loc, C_i) = d(2,6) + d(6,4) + d(4,2) = 2 + 2 + 2 = 6$.

Theorem 1 *The average number of MI local links that must be traversed in a route in SCC_n is:*

$$\overline{MI(loc)} = \frac{(n-1)\left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor}{n} \tag{9}$$

Proof: The average number of local links that must be traversed between two adjacent lateral links is:

$$\overline{d(loc)} = \frac{\sum_{i=3}^{n} d(i,2)}{n-2} = \frac{\left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor}{n-2} \tag{10}$$

The average number of local links that must be traversed in the execution of an *r*-cycle $C_i = (i_0 \ \ldots \ i_{r-1})$ is:

$$\overline{MI(loc, C_i)} = \begin{cases} \overline{d(loc)} \cdot (r-2), & \text{if } i_0 = 1 \\ \overline{d(loc)} \cdot r, & \text{if } i_0 \neq 1 \end{cases} \tag{11}$$

Over all $n!$ possible permutations of n symbols and for each integer r, $2 \leq r \leq n$, there is a total of $(n-1)!$ *r*-cycles that include symbol 1 ($i_0 = 1$) and $n!/r - (n-1)!$ *r*-cycles that do not include symbol 1 ($i_0 \neq 1$). The average number of *MI* local links over all $n!$ permutations is therefore:

$$\frac{\overline{d(loc)} \cdot \sum_{r=2}^{n}\left((n-1)! \cdot (r-2) + \left(\frac{n!}{r} - (n-1)!\right) \cdot r\right)}{n!}$$

$$= \frac{\overline{d(loc)} \cdot (n-1)(n-2)}{n} = \frac{(n-1)\left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor}{n} \quad \square$$

3.4. Average number of *MB* local links

Recall that *MB* local links are needed to move between execution sequences of adjacent *r*-cycles ($2 \leq r \leq n$), to move into the first lateral link, and to move out of the last lateral link in a route in SCC_n.

Theorem 2 *The average number of MB local links that must be traversed in a route in SCC_n, under a random ordering of r-cycles, is:*

$$\overline{MB(loc, rand)} = \left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor \left(\frac{H_n(n-1) - 2}{n^2 - 3n + 2}\right) \tag{12}$$

Proof: Over all $n!$ possible permutations of n symbols and for each integer r, $2 \leq r \leq n$, there is a total of $n!/r$ *r*-cycles. The total number of *r*-cycles of length at least 2 in the $n!$ possible permutations of n symbols is, therefore, $N_r = \sum_{r=2}^{n}(n!/r) = n! \cdot (H_n - 1)$.

The average number of *r*-cycles, $2 \leq r \leq n$, in a permutation of n symbols is $\overline{r} = N_r/n! = H_n - 1$. The average number of *MB* local links that must be traversed between these *r*-cycles is $\overline{MB(loc, mid)} = (\overline{r} - 1) \cdot \overline{d(loc)} = \frac{\left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor (H_n - 2)}{n-2}$.

Let $\langle i, \pi \rangle$ be the source node, and let the first lateral link in the route be i_k, $2 \leq i_k \leq n$. The average number of local links that must be traversed between $\langle i, \pi \rangle$ and $\langle i_k, \pi \rangle$ is $\overline{d(in)} = \frac{1}{n-1} \sum_{i=2}^{n} d(i,2) = \frac{\left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor}{n-1}$.

Note that $\overline{d(in)}$ differs from $\overline{d(loc)}$ (Eq. 10), since to compute $\overline{d(in)}$ we must consider the case $i = i_k$. Similarly, the average number of local links that must be traversed between the last lateral link in the route and the destination node is $\overline{d(out)} = \overline{d(in)}$. Then, the average number of *MB* local links that must be traversed in a route in SCC_n, assuming a random ordering of *r*-cycles, is $\overline{MB(loc, rand)} = \overline{d(in)} + \overline{MB(loc, mid)} + \overline{d(out)}$. The theorem follows. □

As described in Sec. 4, a properly designed routing algorithm can optimize the ordering of the *r*-cycles and reduce the average number of *MB* local links further below the value provided by a random ordering of *r*-cycles (Eq. 12). The average number of *MB* local links, considering that the shortest route between any two nodes of an SCC graph is determined by a minimal routing algorithm, is therefore bounded by:

$$\overline{MB(loc)} \leq \overline{MB(loc, rand)} \tag{13}$$

3.5. Average distance in the SCC graph

Theorem 3 *The average distance of SCC_n is bounded by:*

$$\overline{\phi(SCC_n)} \leq n + H_n + \frac{2}{n} - 4 + \left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor \left(\frac{n^2+1}{n^2-n} + \frac{H_n - 2}{n-2} \right) \tag{14}$$

Proof: Follows directly from Eqs. 5, 6, 9, 12 and 13. □

4. Routing algorithms in the SCC graph

4.1. Ordering of *r*-cycles

Routing between two nodes $\langle i_s, \pi_s \rangle$ and $\langle i_d, \pi_d \rangle$ in SCC_n is equivalent to routing from $\langle i_s, \pi_{ds} \rangle$ to $\langle i_d, \pi_0 \rangle$, where $\pi_{ds} = \pi_d^{-1} \cdot \pi_s$, $\pi_0 = 123\ldots n$, and π_d^{-1} is the *inverse* or *reciprocal* of permutation π_d [1, 10].

Let $P(\ell_1 \mapsto \ell_f)$ denote a route from from $\langle i_s, \pi_s \rangle$ to $\langle i_d, \pi_d \rangle$ in SCC_n, which traverses a sequence of f lateral links $R(\ell_1 \mapsto \ell_f) = (\ell_1, \ell_2, \ldots, \ell_f)$. The total cost of $P(\ell_1 \mapsto \ell_f)$ is given with:

$$|P(\ell_1 \mapsto \ell_f)| = f + d(i_s, \ell_1) + \textstyle\sum_{j=1}^{f-1} d(\ell_j, \ell_{j+1}) + d(\ell_f, i_d) \tag{15}$$

Depending on the order chosen to execute the *r*-cycles in π_{ds}, different routes $P(\ell_1 \mapsto \ell_f)$ are produced. As explained in Sec. 3, a common feature to any of these routes is that they all have the same number of lateral links (lat) and *MI* local links ($MI(loc)$). Finding the shortest route from $\langle i_s, \pi_s \rangle$ to $\langle i_d, \pi_d \rangle$ is therefore a matter of choosing an *r*-cycle ordering which minimizes the number of *MB* local links ($MB(loc)$). A routing algorithm which achieves this goal is given in Subsec. 4.4. Non-minimal (but simpler) routing algorithms are presented in Subsecs. 4.2 and 4.3.

To illustrate the different cost components in a route, and how they are affected by the order chosen to execute the *r*-cycles, assume routing from node $\langle 3, 34125 \rangle$ to node $\langle 2, 12345 \rangle$ in SCC_5. A route along the sequence $R(2 \mapsto 3) = (2, 4, 2, 3)$ contains four lateral links, four *MI* local links, and three *MB* local links (i.e., $|P(2 \mapsto 3)| = 4 + 4 + 3 = 11$). However, if the sequence of lateral links $R(3 \mapsto 2) = (3, 2, 4, 2)$ is used, a route with four lateral links, four *MI* local links, and one *MB* local link results (i.e., $|P(3 \mapsto 2)| = 4 + 4 + 1 = 9$).

In some cases, the number of *MB* local links in a route from $\langle i_s, \pi_s \rangle$ to $\langle i_d, \pi_d \rangle$ can be further reduced by *interleaving* (rather than executing separately) the *r*-cycles in π_{ds}. For example, some possible sequences of lateral links from supernode $\pi_{ds} = 23154 = (1\ 2\ 3)(4\ 5)$ to supernode $\pi_0 = 12345$ in SCC_5 are (2, 3, 4, 5, 4), (2, 3, 5, 4, 5), (4, 5, 4, 2, 3), (5, 4, 5, 2, 3), (2, 4, 5, 4, 3) and (2, 5, 4, 5, 3). The last two of these sequences interleave *r*-cycles (1 2 3) and (4 5). All of the routing algorithms presented in this paper account for the possibility of interleaving *r*-cycles.

4.2. Random routing algorithm

A simple routing algorithm for SCC_n consists of choosing a random order to execute the *r*-cycles in π_{ds}. Particularly, a possible algorithm that can be used for this purpose is the routing algorithm of the star graph [7]:

Algorithm 1 (Non-deterministic routing in the star graph):

Repeat until $\pi_{ds} = \pi_0$:

1. If the first symbol in π_{ds} is 1, then exchange it with any symbol not in its correct position.
2. If the first symbol in π_{ds} is $x \neq 1$, then either exchange it with the symbol at position x, or exchange it with any symbol in an *r*-cycle of length at least two, other than the *r*-cycle containing x.

Algorithm 1 requires at most $c + m$ steps of complexity $O(1)$ each, and therefore its complexity is $O(c + m)$, or $O(n)$, since $0 \leq c \leq \lfloor n/2 \rfloor$ and $1 \leq m \leq n$.

4.3. Greedy routing algorithm

A simple approach to minimizing the number of *MB* local links in the route between nodes $\langle i_s, \pi_s \rangle$ and $\langle i_d, \pi_d \rangle$ consists of using a greedy algorithm. Such an algorithm uses the following data structures and variables:

- $\mathcal{S}_c$ – the set of *r*-cycles of length at least 2 in π_{ds}.
- $\mathcal{S}_s$ – a subset of the symbols of π_{ds}, such that: 1) if $(1\ i_1\ \ldots\ i_{r-1})$ is an *r*-cycle of π_{ds}, $2 \leq r \leq n$, then $i_1 \in \mathcal{S}_s$ and $1, i_2, \ldots, i_{r-1} \notin \mathcal{S}_s$, and 2) if $(i_0\ \ldots\ i_{r-1})$ is an *r*-cycle of π_{ds}, $2 \leq r \leq n$, such that $i_0, \ldots, i_{r-1} \neq 1$, then $i_0, \ldots, i_{r-1} \in \mathcal{S}_s$.
- i_j – an integer variable initialized to $i_j = i_s$.

Algorithm 2 (Greedy routing in the SCC graph):

1. If $\pi_{ds} = \pi_0$, then route inside the supernode and exit.
2. Identify the *r*-cycles of length at least 2 that exist in π_{ds}, and initialize $\mathcal{S}_c$, $\mathcal{S}_s$, and i_j.

3. Choose a symbol $i_\alpha \in \mathcal{S}_s$ such that $d(i_j, i_\alpha)$ is minimal. Let C_a be the r-cycle that contains symbol i_α. Once i_α is chosen, make $i_j = i_\alpha$.

4. If C_a has the form $(1\ i_\alpha\ i_\beta\ \ldots\ i_{r-1})$, then make $\mathcal{S}_c = \mathcal{S}_c - \{(1\ i_\alpha\ i_\beta\ \ldots\ i_{r-1})\} + \{(1\ i_\beta\ \ldots\ i_{r-1})\}$ and $\mathcal{S}_s = \mathcal{S}_s - \{i_\alpha\} + \{i_\beta\}$. Otherwise, make $\mathcal{S}_c = \mathcal{S}_c - \{C_a\}$ and $\mathcal{S}_s = \mathcal{S}_s - \{symbols(C_a)\}$, where $symbols(C_a)$ denotes a function that returns the set of symbols in r-cycle C_a.

5. Repeat Steps 3 and 4 until $\mathcal{S}_c = \emptyset$.

The greedy approach used by Alg. 2 consists of choosing the r-cycle that has the minimum distance from i_j as the next one to be executed. If the selected r-cycle C_a includes symbol 1, then only the first lateral link of C_a is taken, which allows for an interleaved execution of that r-cycle. If C_a does not include symbol 1, then C_a is executed completely. The complexity of the greedy routing algorithm is $O(cm)$, or $O(n^2)$ since $0 \le c \le \lfloor n/2 \rfloor$ and $0 \le m \le n$. The ordering of r-cycles chosen by this algorithm, however, may not produce a minimal route.

4.4. Minimal routing algorithm

We now present a minimal routing algorithm which finds the shortest route between a pair of nodes $\langle i_s, \pi_s \rangle$ and $\langle i_d, \pi_d \rangle$ in SCC_n. The output of the algorithm consists of a sequence of lateral links $R(\ell_1 \mapsto \ell_f)$, for which $|P(\ell_1 \mapsto \ell_f)|$ is minimal (Eq. 15). We note that an earlier version of our minimal routing algorithm appeared in [10]. The algorithm we present here improves that of [10] in two ways: 1) it employs more selective heuristics to further constrain the search space generated by the algorithm, and 2) it accounts for the possibility of interleaving r-cycles, which is not possible with the algorithm in [10].

The algorithm performs a depth-first search on a weighted tree structure. The tree is built by expanding at each step only those r-cycle orderings that seem to result in a minimal number of local links. Although the search tree can virtually examine all possible r-cycle orderings, including interleaved r-cycles, its size is significantly constrained in our algorithm. To guarantee that a minimal route is always found, backtracking is used to enable expansion of previous r-cycle orderings that seem to be better than the most recently expanded orderings.

In the following discussion, we use the term *vertex* to refer to an element of the search tree. In addition, we use the term *edge* to refer to the logical connection between vertices in the search tree, which is usually implemented with pointers or some form of indexing. The following data structures are stored within each vertex v_i of the search tree and are used by the algorithm:

- $\langle \ell_i, \pi_i \rangle$ – the label of the node reached so far by the routing algorithm.

- $\mathcal{B}_i$ – a subset of the symbols of π_i, such that: 1) if $(1\ i_1\ \ldots\ i_{r-1})$ is an r-cycle of π_i, $2 \le r \le n$, then $i_1 \in \mathcal{B}_i$ and $1, i_2, \ldots, i_{r-1} \notin \mathcal{B}_i$, and 2) if $(i_1\ \ldots\ i_{r-1})$ is an r-cycle of π_i, $2 \le r \le n$, such that $i_1, \ldots, i_{r-1} \ne 1$, then $i_1, \ldots, i_{r-1} \in \mathcal{B}_i$.

  The symbols in $\mathcal{B}_i$ represent all possible lateral links that can be selected by the routing algorithm while expanding the search tree from a given vertex v_i. For convenience, we define a function $bsymbols$ to generate $\mathcal{B}_i$ from π_i, such that $\mathcal{B}_i = bsymbols(\pi_i)$.

- $\mathcal{F}_i$ – a subset of the symbols of π_i, such that: 1) if $(1\ i_1\ \ldots\ i_{r-1})$ is an r-cycle of π_i, $2 \le r \le n$, then $i_{r-1} \in \mathcal{F}_i$ and $1, i_1, \ldots, i_{r-2} \notin \mathcal{F}_i$, and 2) if $(i_0\ \ldots\ i_{r-1})$ is an r-cycle of π_i, $2 \le r \le n$, such that $i_0, \ldots, i_{r-1} \ne 1$, then $i_0, \ldots, i_{r-1} \in \mathcal{F}_i$.

  The symbols in $\mathcal{F}_i$ represent all lateral links that can be possibly selected by the routing algorithm to enter supernode π_0 (i.e., all possible r-cycle orderings that can be selected from a given vertex v_i necessarily end with a lateral link $\ell_f \in \mathcal{F}_i$). For convenience, we define a function $fsymbols$ to generate $\mathcal{F}_i$ from π_i, such that $\mathcal{F}_i = fsymbols(\pi_i)$.

- L_i – the number of local links used so far by the routing algorithm in the route from $\langle i_s, \pi_{ds} \rangle$ to $\langle \ell_i, \pi_i \rangle$.

- M_i – an estimate of the minimum number of local links that may be needed to reach node $\langle i_d, \pi_0 \rangle$ from node $\langle i_s, \pi_{ds} \rangle$, using the route already constructed by the algorithm up to the intermediate node $\langle \ell_i, \pi_i \rangle$. For convenience, we define a function dubbed $minloc$, which computes M_i as follows:

  $$Mi = minloc(L_i, \ell_i, \pi_i, i_d) = L_i + \min(d(\ell_i, b_i)) + \sum^{C_i \in \pi_i} MI(loc, C_i) + \min(d(f_i, i_d)), \quad (16)$$

  where $b_i \in \mathcal{B}_i$ and $f_i \in \mathcal{F}_i$.

  Note that $minloc$ is computed under the optimistic assumption that the route from $\langle \ell_i, \pi_i \rangle$ to $\langle i_d, \pi_0 \rangle$ selects the best possible lateral links in $\mathcal{B}_i$ and $\mathcal{F}_i$. In addition, the summation term which computes the number of local links needed to execute all r-cycles $C_i \in \pi_i$ (see Eq. 8) assumes that an optimal r-cycle ordering requiring no local links to move from one r-cycle to the next can be found by the routing algorithm.

- e_i – an enable/disable bit which indicates whether or not the tree should be expanded from vertex v_i.

In addition, the tree structure generated by the minimal routing algorithm has the following characteristics:

- The search tree has at most $lat + 2$ levels, with lat being given by Eq. 4. We number levels from 0 to $lat + 1$, starting from the root level.

- Let v_i be the parent of a vertex v_i' in the search tree. Let $\{\langle \ell_i, \pi_i \rangle, \mathcal{B}_i, \mathcal{F}_i, L_i, M_i, e_i\}$ and $\{\langle \ell_i', \pi_i' \rangle, \mathcal{B}_i', \mathcal{F}_i', L_i', M_i', e_i'\}$ denote the data stored in v_i and v_i', respectively. The weight of the edge (v_i, v_i') corresponds to the number of local links that are required to route from $\langle \ell_i, \pi_i \rangle$ to $\langle \ell_i', \pi_i' \rangle$ in SCC_n and is given by $d(\ell_i, \ell_i')$. Hence, $L_i' = L_i + d(\ell_i, \ell_i')$.

  Note that routing from $\langle \ell_i, \pi_i \rangle$ to $\langle \ell_i', \pi_i' \rangle$ also requires one lateral link if $\pi_i \neq \pi_i'$, and zero lateral links otherwise. Since the number of lateral links in a route from $\langle i_s, \pi_{ds} \rangle$ to $\langle i_d, \pi_0 \rangle$ can be computed a priori (Eq. 4), the routing algorithm focuses on accounting for the local links only.

- Vertices located at level $lat + 1$ in the tree have $\langle \ell_i, \pi_i \rangle = \langle i_d, \pi_0 \rangle$, $\mathcal{B}_i = \mathcal{F}_i = \emptyset$ and $M_i = minloc(L_i, i_d, \pi_0, i_d) = L_i$. Vertices located at level lat have $\langle \ell_i, \pi_i \rangle = \langle \ell_f, \pi_0 \rangle$ (with ℓ_f being the lateral link used to enter supernode π_0), $\mathcal{B}_i = \mathcal{F}_i = \emptyset$, and $M_i = minloc(L_i, \ell_f, \pi_0, i_d) = L_i + d(\ell_f, i_d)$.

- The backtracking mechanism is triggered by comparing the estimated minimum number of local links (M_i) stored in the most recently generated child vertices with a global variable referred to as T. This variable is updated whenever a backtracking procedure occurs, meaning that the minimum number of local links that is required in the route from $\langle i_s, \pi_{ds} \rangle$ to $\langle i_d, \pi_0 \rangle$ is actually greater than the previous value of T. The search becomes more selective as T increases, which not only limits the width of the search tree, but also makes the backtracking mechanism less likely to be triggered again.

Given the definitions above, the minimal routing algorithm for the SCC graph follows :

Algorithm 3 (Minimal routing in the SCC graph):

1. If $\pi_{ds} = \pi_0$, then route inside the supernode and exit.
2. Create a root vertex with $\langle \ell_i, \pi_i \rangle = \langle i_s, \pi_{ds} \rangle$, $\mathcal{B}_i = bsymbols(\pi_{ds})$, $\mathcal{F}_i = fsymbols(\pi_{ds})$, $L_i = 0$, $M_i = minloc(0, i_s, \pi_{ds}, i_d)$ and $e_i =$ ON. Also, initialize T with the value $T = minloc(0, i_s, \pi_{ds}, i_d)$.
3. Generate child vertices for all enabled vertices, such that the label ℓ_i' for each child corresponds to exactly one of the symbols stored in the set $\mathcal{B}_i$ of each parent vertex. Set $e_i =$ OFF at each recently expanded parent vertex. Also, obtain permutation π_i' for each child vertex by swapping the 1st and the ℓ_i'th symbols of π_i, and make $\mathcal{B}_i' = bsymbols(\pi_i')$, $\mathcal{F}_i' = fsymbols(\pi_i')$, $L_i' = L_i + d(\ell_i, \ell_i')$, $M_i' = minloc(L_i', \ell_i', \pi_i', i_d)$. Enabled vertices located at level lat of the search tree must be expanded similarly. However, they generate a single child with $\ell_i' = i_d$, $\pi_i' = \pi_i$, $\mathcal{B}_i' = \emptyset$, $\mathcal{F}_i' = \emptyset$, $L_i' = L_i + d(\ell_i, i_d)$, and $M_i' = L_i'$. In any case, a child vertex is enabled with $e_i' =$ ON if $M_i' \leq T$. Otherwise, we set $e_i' =$ OFF.
4. If a child vertex has $\langle \ell_i', \pi_i' \rangle = \langle i_d, \pi_0 \rangle$ and $e_i' =$ ON, then a minimal route has been found. The optimal sequence of lateral links $R(\ell_1 \mapsto \ell_f)$ can be obtained in reverse order by backing up towards the root of the tree and listing the value ℓ_i stored in each vertex located between the lat^{th} and the 1st levels. Once $R(\ell_1 \mapsto \ell_f)$ has been obtained, exit the algorithm.
5. If none of the enabled child vertices has $\langle \ell_i', \pi_i' \rangle = \langle i_d, \pi_0 \rangle$, go to Step 3.
6. If there are no enabled child vertices, do a backtracking search in the tree. Among all existing child vertices, select those with the smallest value of M_i and set T to this value. Also, enable the selected nodes and go to Step 4.

The height of the search tree is $O(n)$, since its maximum value is $\phi(S_n) + 2 = \lfloor 3\,(n-1)/2 \rfloor + 2$. A worst-case analysis of the width of the search tree can be done under the following pessimistic assumption: considering that all possible orderings of r-cycles in permutation π_{ds} are examined by Alg. 3, the lowest level in the search tree would have at most $m!$ vertices. This is due to the fact that there are at most $m!$ possible ways to move the m misplaced symbols in π_{ds} to their correct positions, using the minimum number of lateral links given by Eq. 4. In practice, the constraints placed on the number of vertices by the heuristics of Alg. 3 (i.e., the estimated minimum number of local links M_i) limit the width of the search tree considerably. Simulations carried out for $4 \leq n \leq 9$ revealed that a very small number of vertices is enabled at each step, which makes the maximum width of the tree virtually proportional to m. Figure 3 illustrates an example of the search tree constructed by Alg. 3.

The main computations incurred upon creation of a vertex of the search tree refer to $\mathcal{B}_i'$, $\mathcal{F}_i'$ and M_i'. Fortunately, each of these computations can be accomplished in $O(1)$ time by using the corresponding values $\mathcal{B}_i$, $\mathcal{F}_i$ and M_i that are stored in the parent vertex, and taking into account the differences in the r-cycle structures of permutations π_i and π_i'.

The reasoning above results in a worst-case complexity of $O(m!n)$. As explained above, such computational requirements were not observed during simulations of the minimal algorithm. The potential need for backtracking searches in the tree, added to fact that the maximum width of the tree is in practice proportional to m, results in a complexity of $O(mn^2)$, on the average (or $O(n^3)$, since $0 \leq m \leq n$).

5. Simulation results

The performance of routing algorithms for SCC_n was evaluated with simulation programs which compute the route of all $(n-1)n!$ nodes of the graph to the identity. The routing algorithms that were tested are: 1) a random routing algorithm that generates all possible routes to the identity with equal probability, which is based on Alg. 1, 2) Alg. 2, and 3) Alg. 3. The simulations were carried out for $3 \leq n \leq 9$. A log of worst-case routes that may result from the random routing algorithm was also made.

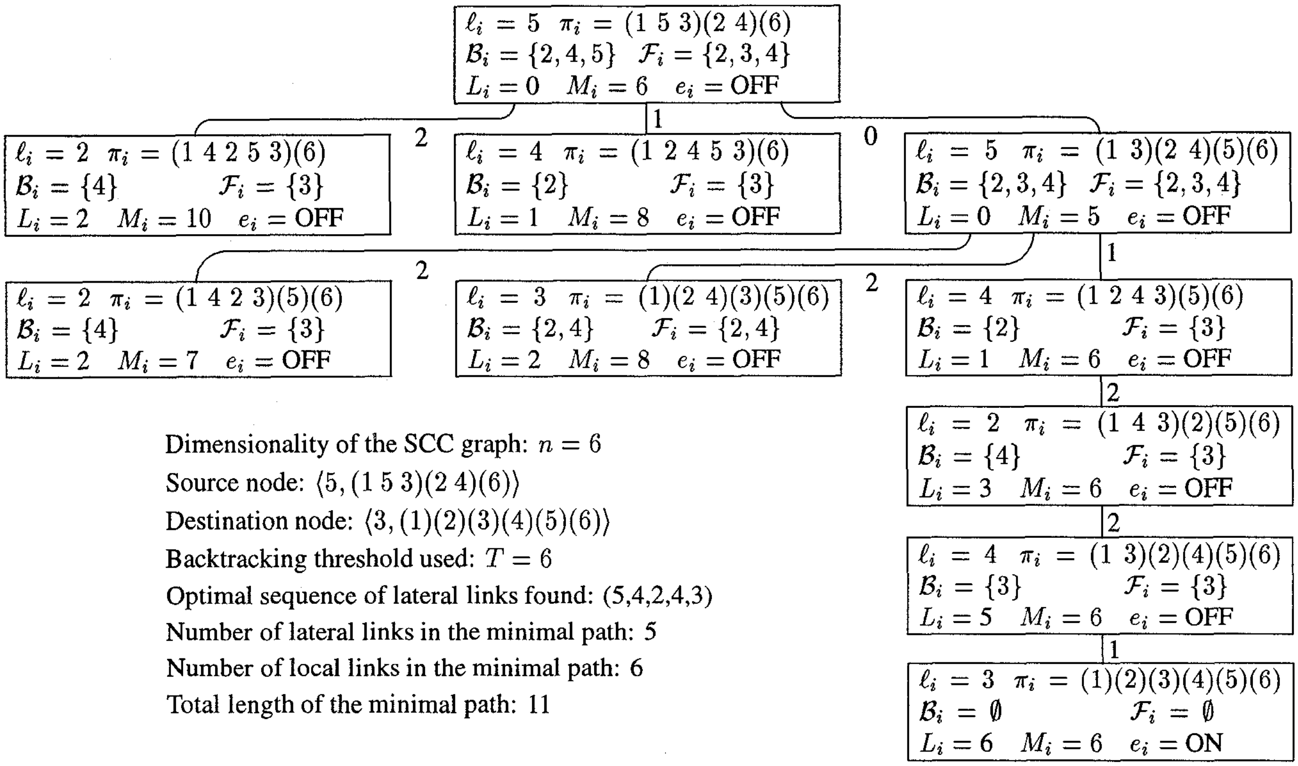

Figure 3. Example of search tree used for minimal routing in SCC_n

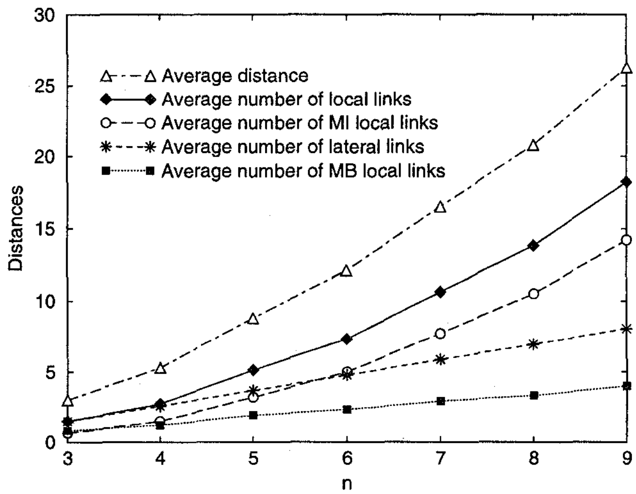

Figure 4. Av. distances under minimal routing

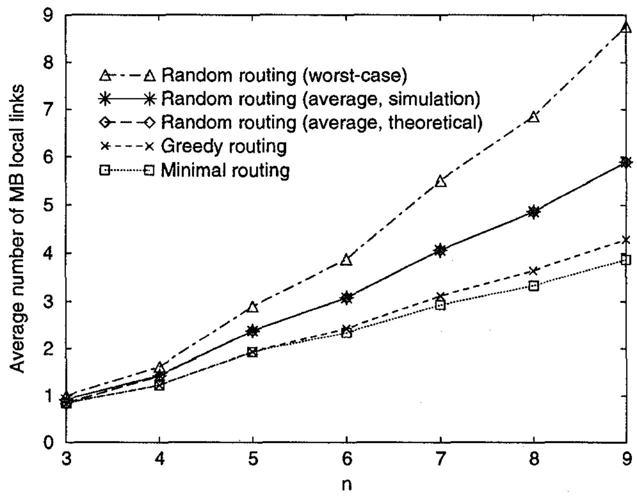

Figure 5. $\overline{MB(loc)}$ vs. routing algorithms

Table 1 and Fig. 4 show the simulation results obtained with the minimal routing algorithm. Values for $\overline{lat}$ and $\overline{MI(loc)}$ match exactly the theoretical values provided by Eqs. 6 and 9. Also, the simulation results obtained for $\overline{MB(loc)}$ under a minimal routing algorithm are closely bounded by Eq. 12.

As expected, only the average number of MB local links varied among the different routing algorithms that were tested. Fig. 5 compares simulation results for $\overline{MB(loc)}$. Note that the results for the random routing algorithm are very close to the theoretical values provided by Eq. 12. The model used to derive that equation seems to result in an error proportional to $1/n!$, which is negligible considering that Eq. 12 is still a close upper bound for $\overline{MB(loc)}$. As expected, both the greedy and the minimal routing algorithm outperform the random routing algorithm, as far as the av-

n	3	4	5	6	7	8	9
Graph size ($(n-1) \cdot n!$)	12	72	480	3,600	30,240	282,240	2,903,040
Graph diameter ($\phi(SCC_n)$)	6	8	16	19	31	34	50
Average number of lateral links ($\overline{lat}$)	1.500	2.583	3.683	4.783	5.879	6.968	8.051
Average number of MI local links ($\overline{MI(loc)}$)	0.667	1.500	3.200	5.000	7.714	10.500	14.222
Average number of MB local links ($\overline{MB(loc)}$)	0.833	1.222	1.925	2.337	2.924	3.334	3.873
Average number of local links ($\overline{loc}$)	1.500	2.722	5.125	7.337	10.638	13.834	18.096
Average distance ($\overline{\phi(SCC_n)}$)	3.000	5.306	8.808	12.121	16.517	20.802	26.147

Table 1. Average distance of SCC graphs under minimal routing

erage number of MB local links is concerned. Also observe that, for $3 \leq n \leq 4$, the greedy routing algorithm performs as well as the minimal routing algorithm. Besides, our results indicate that the performance of these algorithms is quite similar for $5 \leq n \leq 9$, which makes the less complex greedy routing algorithm particularly attractive.

Average costs of paths produced by the three routing algorithms are summarized in Table 2. The random routing algorithm has a complexity of $O(n)$ and performs reasonably well on the average. Utilization of such an algorithm may, however, result in variations in the average cost of routes up to the worst-case values shown in Table 2.

n	Minimal rout.	Greedy rout.	Random routing		
			Theor.	Simul.	Worst-case
3	3.000	3.000	3.000	3.084	3.167
4	5.306	5.305	5.500	5.514	5.694
5	8.808	8.812	9.261	9.264	9.775
6	12.121	12.215	12.858	12.858	13.662
7	16.517	16.707	17.660	17.660	19.100
8	20.802	21.109	22.332	22.332	24.324
9	26.147	26.570	28.168	28.168	31.043

Table 2. Average costs vs. routing algorithms

Figure 6 shows distribution curves comparing the three routing algorithms in the case of an SCC_9 graph. A point (D_I, N_I) in one of these curves indicates that the corresponding routing algorithm will compute a route of cost D_I to the identity for N_I nodes in the SCC graph. The average distribution for the random routing algorithm is shown, but the results for that algorithm may actually vary from the minimal to the worst-case distribution curves due to the nondeterministic nature of the algorithm. It is also interesting to observe that the greedy routing algorithm provides a distribution curve which is close to that of the minimal routing algorithm, presenting however a smaller complexity.

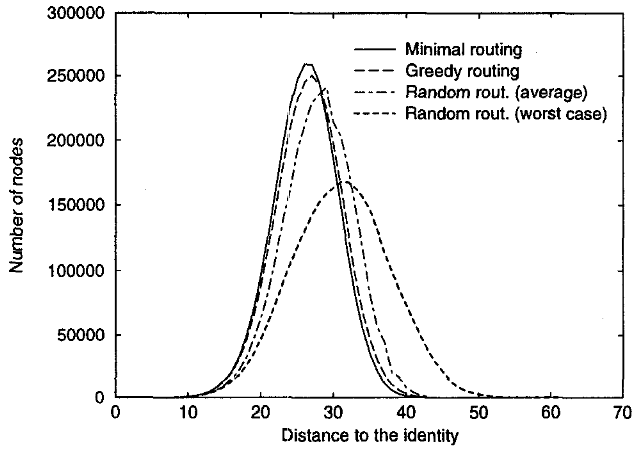

Figure 6. $D_I \times N_I$ dist. curves for SCC_9

6. Considerations on wormhole routing

In this section, we briefly describe how the algorithms presented in the paper can be combined with wormhole routing [6], which is a popular switching technique used in parallel computers.

All three algorithms can be used with wormhole routing, when implemented as *source-based routing* algorithms [11]. In source-based routing, the source node selects the entire path before sending the packet. Because the processing delay for the routing algorithm is incurred only at the source node, it adds only once to the communication latency, and can be viewed as part of the start-up latency. Source-based routing, however, has two disadvantages: 1) each packet must carry complete information about its path in the header, which increases the packet length, and 2) the path cannot be changed while the packet is being routed, which precludes incorporating adaptivity into the routing algorithm.

Distributed routing eliminates the disadvantages of source-based routing by invoking the routing algorithm in each node to which the packet is forwarded [11]. Thus, the decision on whether a packet should be delivered to the local processor or forwarded on an outgoing link is done locally by the routing circuit of a node. Because the routing algorithm is invoked multiple times while a packet is being routed, the routing decision must be taken as fast as possible. From this viewpoint, it is important that the routing algorithm can be easily and efficiently rendered in hardware, which favors the random routing algorithm over the greedy and minimal routing algorithms.

Besides being the most complex algorithm discussed in this paper, the minimal routing algorithm includes a feature which precludes its distributed implementation in association with wormhole routing, namely its backtracking mechanism. Distributed versions of the random and greedy algorithms, however, can be used in combination with wormhole routing. A near-minimal distributed routing algorithm which supports wormhole routing can be obtained by removing the backtracking mechanism from Alg. 3. Such an algorithm is likely to have computational complexity and average cost that lie between those of the greedy and the minimal routing algorithm.

Due to its non-deterministic nature, the random routing algorithm also seems to be a good candidate for SCC networks employing distributed adaptive routing [11]. Adaptivity is desirable, for example, if the routing algorithm must dynamically respond to network conditions such as congestion and faults. Some degree of adaptivity is also possible in the greedy and minimal routing algorithms, which in some cases can decide between paths of equal cost.

7. Conclusion

This paper compared the average cost and the complexity of three different routing algorithms for the SCC graph. We divided routes into three components (lateral links, MI local links and MB local links) and showed that only the number of MB local links may be affected by the routing algorithm being considered. Exact expressions for the average number of lateral links and the average number of MI local links were presented. Also, an upper bound for the average number of MB local links was derived, considering a random routing algorithm. As a result, a tight upper bound on the average distance of the SCC graph was obtained.

Simulation results for a random, a greedy and a minimal routing algorithm were presented and compared with theoretical values. The complexity of the proposed algorithms is respectively $O(n)$, $O(n^2)$, and $O(n^3)$, where n is the dimensionality of the SCC_n graph. The results under minimal routing produce exact numerical values for the average distance of SCC_n, for $3 \leq n \leq 9$.

Results for the greedy algorithm match those of the minimal algorithm for $3 \leq n \leq 4$. The greedy algorithm also performs close to minimality for $5 \leq n \leq 9$, and is an interesting choice due to its $O(n^2)$ complexity. The random routing algorithm has an $O(n)$ complexity and performs fairly well on the average, but may introduce additional MB local links in the route under worst-case conditions.

Finally, we discussed how each of the routing algorithms can be used in association with the wormhole routing switching technique. Directions for future research in this area include an evaluation of requirements for deadlock avoidance (e.g., number of virtual channels).

References

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

[2] M. M. Azevedo, N. Bagherzadeh and S. Latifi, "Broadcasting Algorithms for the Star-Connected Cycles Interconnection Network," *J. Par. Dist. Comp.*, 25, 209-222 (1995).

[3] M. M. Azevedo, N. Bagherzadeh, and S. Latifi, "Embedding Meshes in the Star-Connected Cycles Interconnection Network," to appear in *Math. Mod. and Sci. Comp.*

[4] M. M. Azevedo, N. Bagherzadeh, and S. Latifi, "Fault-Diameter of the Star-Connected Cycles Interconnection Network," *Proc. 28th Annual Hawaii Int'l Conf. Sys. Sci.*, Vol. II, Jan. 3-6, 1995, pp. 469-478.

[5] W.-K. Chen, M. F. M. Stallmann, and E. F. Gehringer, "Hypercube Embedding Heuristics: An Evaluation," *Int'l J. Par. Prog.*, Vol. 18, No. 6, 1989, pp. 505-549.

[6] W. J. Dally and C. I. Seitz, "The Torus Routing Chip," *Dist. Comp.*, Vol. 1, No. 4, 1986, pp. 187-196.

[7] K. Day and A. Tripathi, "A Comparative Study of Topological Properties of Hypercubes and Star Graphs," *IEEE Trans. Par. Dist. Sys.*, Vol. 5, No. 1, Jan. 1994, pp. 31-38.

[8] D. E. Knuth, *The Art of Computer Programming, Vol. 1*, Addison-Wesley, 1968, pp. 73, pp. 176-177.

[9] S. Latifi, "Parallel Dimension Permutations on Star Graph," *IFIP Trans. A: Comp. Sci. Tech.*, 1993, A23, pp. 191-201.

[10] S. Latifi, M. M. Azevedo and N. Bagherzadeh, "The Star-Connected Cycles: A Fixed-Degree Interconnection Network for Parallel Processing," *Proc. Int'l Conf. Par. Proc.*, 1993, Vol. 1, pp. 91-95.

[11] L. M. Ni and P. K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Routing Techniques," *Computer*, Feb. 1993, pp. 62-76.

[12] F. P. Preparata and J. Vuillemin, "The Cube-Connected Cycles: A Versatile Network for Parallel Computation," *Comm. ACM*, Vol. 24, No. 5, May 1981, pp. 300-309.

[13] Y. Saad and M. H. Schultz, "Topological Properties of Hypercubes," *IEEE Trans. Comp.*, Vol. 37, No. 7, July 1988, pp. 867-872.

[14] S. Shoari and N. Bagherzadeh, "Computation of the Fast Fourier Transform on the Star-Connected Cycle Network," to appear in *Comp. & Elec. Engr.*, 1996.

[15] P. Vadapalli and P. K. Srimani, "Two Different Families of Fixed Degree Regular Cayley Networks," *Proc. Int'l Phoenix Conf. Comp. Comm.*, Mar. 28-31, 1995, pp. 263-269.