ACKNOWLEDGMENT

We are grateful to G. Lachaud for showing us [8] at exactly the moment that we were inventing and studying the notion of higher weights for projective systems. The last two authors would also like to thank the United Kingdom Science and Engineering Research Council for its support and the University of Sussex at Brighton for its warm hospitality.

REFERENCES

[1] I. M. Chakravarti, "Families of codes with few distinct weights from singular and nonsingular Hermitian varieties and quadrics in projective geometries and Hadamard difference sets and designs associated with two-weight codes," in *IMA Vol. Appl. Math.*, no. 20, Springer, New York, pp. 35–50, 1990.

[2] G. L. Feng, K. K. Tseng, and V. K. Wei, "On the generalized Hamming weights of several classes of cyclic codes," *IEEE Trans. Inform. Theory*, vol. 38, pp. 1125–1130, May 1992.

[3] T. Helleseth, T. Kløve, and Ø. Ytrehus, "Generalized Hamming weights of linear codes," *IEEE Trans. Inform. Theory*, vol. 38, pp. 1133–1140, May 1992.

[4] J. W. P. Hirschfeld and J. A. Thas, *General Galois Geometries.* Oxford: Oxford University Press, 1991.

[5] D. Nogin, "Generalized Hamming weights for codes on multi-dimensional quadrics," *Problems Inform. Transmission*, to appear.

[6] M. A. Tsfasman and S. G. Vlăduţ, *Algebraic-Geometric Codes.* Dordrecht: Kluwer Academic, 1991.

[7] Z. Wan, "The weight hierarchies of the projective codes from nondegenerate quadrics," *Des. Codes Cryptograph.*, to appear.

[8] V. K. Wei, "Generalized Hamming weights for linear codes," *IEEE Trans. Inform. Theory*, vol. 37, pp. 1412–1418, Sept. 1991.

[9] K. Yang, P. V. Kumar, and H. Stichtenoth, "On the weight hierarchy of geometric Goppa codes," to be published.

The Complexity of Routing in Clos Permutation Networks

David M. Koppelman, *Member, IEEE,*
and A. Yavuz Oruç, *Senior Member, IEEE*

***Abstract*—A lower bound on the amount of information necessary to compute the switch settings for a three-stage Clos network is derived. The bound is derived by considering the effect of a family of permutations, called balanced multiloops, on the settings of the switches of a Clos network. By carefully selecting these permutations, it is proven that there exists at least one switch in each stage whose setting depends upon at least $(k-3)(m/2+1)/2$ assignments in the permutations to be routed for $k \geq 4$ where m is the number of center-stage switches and k is the number of input and output-stage switches. It is also proven that the setting of the input stage depends on at least $m-1$ assignments. Lower bounds on the routing time of a three-stage Clos network on a variety of machine models follow immediately from these results. In particular, any constant fan-in implementation of any routing algorithm for such a network should have a time lower bound of $\Omega(\log mk)$.**

***Index Terms*—Clos network, edge-coloring, information complexity of routing, lower bounds, network routing, permutation network, routing complexity.**

I. INTRODUCTION

This paper examines the information complexity of routing in Clos permutation networks. Such networks are typically formed recursively from smaller networks, and have widely been investigated as connectors in telephone switching [5], [22], [28] and parallel computer systems [6], [10], [23].

Formally, a three-stage Clos permutation network, henceforth to be called a Clos network, is defined in terms of two parameters, m and k where m is the number of inputs to each of the switches in the outer stages, and k is the number of inputs to each of the switches in the center stage [8], [5]. The total number of inputs (outputs) to the network is given by mk and will be denoted by n. The interconnections between consecutive stages are such that there exists exactly one link between every center- and outer-stage switch. Consequently, the parameter k also specifies the number of switches in each of the outer stages while m is the number of switches in the center stage.

Here, *routing* a network refers to determining the mappings of the inputs of its switches from a permutation so as to connect its inputs to its outputs as specified in the permutation. For a network with n inputs and n outputs, a permutation specifies n pairs of inputs and outputs, each of which is called an *assignment.* Let $a_{i,j}$ be the minimum number of assignments that must be examined in order to determine the mapping of input j of switch i in a network for any given permutation. The *information complexity of routing* a network is the largest $a_{i,j}$ over all the inputs of all the switches in the network. A lower bound on the routing time of a network immediately follows from a lower bound on the information complexity of routing

Manuscript received January 23, 1992; revised October 22, 1992. This work was supported in part by the National Science Foundation under Grant CCR-8708864. This paper was presented in part at the Allerton Conference, Urbana, IL, October 1988.

D. M. Koppelman is with the Department of Electrical and Computer Engineering, Louisiana State University, Baton Rouge, LA 70803.

A. Y. Oruç is with the Department of Electrical Engineering and the Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.

IEEE Log Number 9215121.

that network. In particular, if a network has an $\Omega(r)$ information[1] complexity of routing then any constant fan-in implementation of any routing algorithm[2] for that network would have a time lower bound of $\Omega(\log r)$.

Much has been reported on routing Clos networks. Opferman and Tsao–Wu [27], and Waksman [31] described the $\Theta(n \log n)$-time sequential algorithm for an n-input Clos network with binary switches, called the Beneš network [5]. An extension of this algorithm to Clos networks with $2^t \times 2^t$ switches where $t \geq 1$ was given by Andresen [2]. For the Beneš network, Nassimi and Sahni [24] described a parallel routing algorithm which runs on an n-processor computer with a completely connected communication graph in $\Theta(\log^2 n)$ time. For Clos networks with an arbitrary number of inputs, matrix decomposition heuristics [3], [15], [26] facilitate routing with backtracking, which in general result in exponential time. More efficient routing algorithms for these networks were devised by Lev, Pippenger, and Valiant [21] based on edge-coloring bipartite graphs. Their parallel routing algorithm requires $\Theta(\log^2 n \log m)$ time for a three-stage Clos network with $m \times m$ switches in its outer stages, and it takes $\Theta(\log^3 n)$ time for a recursively decomposed Clos network on an EREW PRAM: an n-processor computer in which multiple processors can access any part of a shared memory in unit time, but in which read and write conflicts are not allowed [16]. As shown in [7], [14], other efficient routing algorithms for permutation networks can be obtained by using the edge-coloring algorithms described in [9], [12], [13].

As with other lower bound results, it is important to determine the information complexity of routing Clos networks in order to determine the effectiveness of these routing schemes. We prove that the information complexity of routing a Clos network consisting of $m \times m$ switches in its first and third stages, and $k \times k$ switches in its center stage is at least $(k-3)(m/2+1)/2$ for $k \geq 4$. To establish this result, we examine the dependencies between the assignments in permutation requests and the mappings of the inputs of switches in a three-stage Clos network. A simple fan-in argument then yields the $\Omega(\log_w mk)$ lower bound on the routing time of such a network for any routing model with w fan-in. This lower bound in turn establishes that it takes $\Omega(n + (n/m)\log_w n)$ steps to route a recursively decomposed three-stage Clos network, if its subnetworks are routed in sequence, and $\Omega(\log_m n \log_w n)$ steps if they are routed in parallel. (These bounds are given in greater detail in Section V.)

It is worthwhile to contrast the information complexity of routing Clos networks with the information complexity of routing other networks, in particular, self-routing networks. First, we note that the information complexity of routing a self-routing network is never greater than the maximum number of inputs that can reach a switch in that network. For example, the information complexity of routing the first stage of a unique-path network such as those described in [20] is one since it suffices to examine a single bit of one input of each first-stage switch to determine its setting. When operated as a self-routing network, the information complexity of routing the first stage of the Beneš network is two, since it suffices to examine either one or two inputs to determine the mapping of either input of any switch in that stage [25], [29]. It should be noted that, unlike Clos networks, unique-path and Beneš networks (in self-routing mode) cannot realize all permutations [25], [29]. On the other hand, Batcher's sorting networks [4] can route all permutations, and the information complexity of routing their first stage is only two. The tradeoff in this case is hardware cost; Batcher sorters use $\Theta(n \log^2 n)$ switches whereas Clos networks can be minimized to have $\Theta(n \log n)$ switches.

[1] In expressing complexities, we use the standard complexity notations O, Ω, and Θ; see [17] for definitions.

[2] Here, the fan-in of an implementation of a routing algorithm refers to the maximum number of inputs each elementary device can have in that implementation.

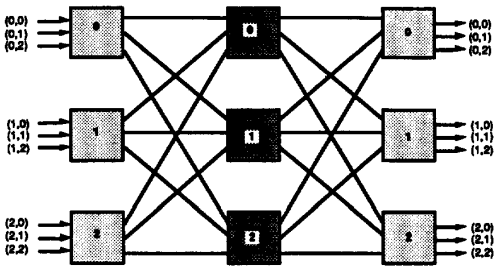

Fig. 1. A 9-input Clos permutation network.

In all three networks, the fact that the information complexity of routing the first stage is a constant and also that these networks have a recursive structure explain why they are very easy to route. In contrast, the main result of this paper shows that the information complexity of routing the first stage (as well as the second and third stages) of a Clos network is about as large as the number of inputs to the network, and this explains why these networks cannot be routed as they are traversed.

The remainder of the paper is organized as follows. Section II states preliminary facts needed in subsequent sections. Section III introduces the notions of multiloops and balanced multiloops, and Section IV uses these to obtain the claimed information complexity lower bound. Section V uses this lower bound to determine the time complexities of various routing schemes for Clos networks. In Section VI extant routing algorithms are compared to the bounds. The paper is concluded in Section VII.

II. Preliminary Facts

A three-stage Clos network will be defined in terms of two parameters, m and k where m is the number of inputs to each of the switches in the outer stages, and k is the number of inputs to each of the switches in the center stage [8], [5]. The total number of inputs (outputs) to the network is mk and will be denoted by n. Note that k is also the number of switches in each of the outer stages while m is also the number of switches in the center stage. The interconnections between consecutive stages are such that there exists exactly one link between every center- and outer-stage switch. For example, a Clos network appears in Fig. 1 for $n = 9$, and $m = k = 3$.

Input and output terminals to the network will be labeled by pairs (i, j); $0 \leq i \leq k-1$, $0 \leq j \leq m-1$ where (i, j) denotes the jth input or output (depending on context) of switch i. A permutation π to be realized by the network will be specified in the form of $[(i, j), \pi(i, j)]$, $0 \leq i \leq k-1$, $0 \leq j \leq m-1$ where $\pi(i, j)$ is the output to which input (i, j) is mapped. When we do not wish to emphasize to which switches inputs or outputs belong, we will denote them by integers $0, 1, \ldots, n-1$ and express π as $[i, \pi(i)]$, $0 \leq i \leq n-1$. Permutations for the input, center, and output stages will be denoted by π_I, π_M, π_O, respectively.

Our model of routing a Clos network, to be called a *router*, will be a black box that receives a permutation π in the form of a set of assignments and returns a set of settings for the switches in the network so as to realize π. These settings, for each stage of the network, may be viewed as a permutation π_s, $s \in \{I, M, O\}$. For a fixed x, y, and s, $\pi_s(x, y)$ specifies the image of input y of switch x in stage s, i.e., the output of switch x in stage s to which its input y is mapped.

In determining switch settings, we will be interested only in the minimum number of assignments that the router must use, i.e., the information complexity of setting switches, rather than the actual computation that the router engages in to determine the settings. Once such an information lower bound is obtained then the structure of the router can be brought to bear on that bound to determine a lower bound on its routing time.

The information bound on routing a three-stage Clos network will be derived by finding *dependencies* between assignments in π and settings in permutations π_s.

Definition 1: Two permutations π_1 and π_2 are said to differ on a set of inputs if their assignments on those inputs are different. For example, $\pi_1 = [0, 4], [1, 3], [2, 1], [3, 2], [4, 5], [5, 0]$ and $\pi_2 = [0, 4], [1, 3], [2, 1], [3, 2], [4, 0], [5, 5]$ differ on inputs 4 and 5. □

Definition 2: Let π be a permutation to be realized by a Clos network and Δ be a subset of inputs of stage $s \in \{I, M, O\}$. Define $V_\Delta(\pi, s)$ to be the set of all mappings of the subset of inputs Δ of stage s which do not prevent the Clos network from realizing π. □

The set $V_\Delta(\pi, s)$ will be used to define the dependencies between the assignments in π and settings in π_s.

Definition 3: Let A be a set of assignments and let Δ and s be defined as above. The mappings of inputs in Δ are said to be dependent upon A (briefly, Δ depends on A) if there exist two permutations π_1 and π_2 which differ only in inputs which appear in the assignments that belong to A and such that $V_\Delta(\pi_1, s) \cap V_\Delta(\pi_2, s) = \phi$. □

Because $V_\Delta(\pi_1, s) \cap V_\Delta(\pi_2, s) = \phi$ the router must provide different mappings for the inputs in Δ under π_1 and π_2 if the network is to be set up properly. To distinguish π_1 and π_2 the router must examine the assignments in A. That is, if Δ depends upon A, a router which computes Δ must base its computation on the part of the permutation specified by A.

Although dependencies are defined above for a set of assignments, it will be useful to define them for an individual assignment. This cannot be done directly using the above definition because the permutations π_1 and π_2 cannot be found when A contains exactly one element, since if two permutations differ they must differ in at least two places. Despite this fact, the dependency of a switch setting on a subset of assignments can be inferred indirectly as stated in the following.

Proposition 1: If the mappings for a set of inputs Δ of a set of switches depend upon a set of assignments A, then they depend on at least one of the assignments in A. □

The set Δ may depend on A without having to depend on all the assignments in A. That is, we may have two permutations π_1 and π_2 which differ only on assignments in A and for which the mappings of the inputs in Δ may be disjoint, but no two such permutations may exist for some subset of A. The following results establish that, under certain conditions, we may determine the number of "element-wise" dependencies from "subset-wise" dependencies.

Lemma 1: If the mappings for a set of inputs Δ of a set of switches depend upon all subsets of assignments of cardinality r of a set of assignments A then they depend upon at least $|A|-r+1$ assignments.

Proof: The largest set of assignments which Δ could not depend on, consistent with the Lemma, is one with cardinality $r - 1$, so Δ depends on at least $|A| - r + 1$ assignments. □

Lemma 2: Given an input δ of a switch and z disjoint sets of assignments $A_0, A_1, \ldots, A_{z-1}$, if the mapping of δ depends upon $A_i \cup A_j$ for all $0 \leq i < j < z$ then it depends upon either all, or all but one of the A_i.

Proof: Suppose the mapping of δ does not depend on one of the sets, say A_0. Then it must depend on A_i; $1 \leq i \leq z - 1$ since it depends on $A_0 \cup A_i$ for all i; $1 \leq i \leq z - 1$. □

These facts, combined with the results of Section III, will be used in Section IV to find the minimum number of assignments necessary to determine the stage settings for a three-stage Clos network for some permutations. Once the dependencies are determined the lower bound on the time it takes to route a 3-stage Clos network immediately follows.

III. Permutations as Multigraphs and Multiloops

To find the dependencies, permutations with special properties will be described. These permutations will be represented as bipartite multigraphs. Let $\mathbf{Z}_k = \{0, 1, \ldots, k - 1\}$. We shall write (I, M, O) to denote a bipartite multigraph with two disjoint sets of vertices, $I = \mathbf{Z}_k$, and $O = \mathbf{Z}_k$, and a multiset[3] of edges $M \subset (I \times O)^m$ connecting the two sets of vertices where $(I \times O)^m$ is a multiset containing $I \times O$ m times. The elements in I will be referred to as *left vertices,* and represent the input-stage switches while the elements in O will be referred to as *right vertices* and represent the output-stage switches. An edge between left vertex i and right vertex j will be denoted by $[i, j]$. For each permutation π, the set M contains r edges $[i, j]$ iff $\pi(i, x) = (j, y)$ for exactly r distinct pairs of integers (x, y) where x (y) is an input (output) of switch i (j) in the input (output) stage.

Thus, when a left vertex i is connected to a right vertex j by r edges the connection represented has exactly r inputs which are mapped from input-stage switch i to output-stage switch j under the corresponding permutation. Each (x, y) pair specifies an input x of input stage switch i which is connected to output y of output stage switch j. Furthermore, it can be shown that for any given m and k, the bipartite multigraph representation of each permutation for an n-input Clos network is unique, while many permutations can have the same bipartite multigraph representation.

For any given permutation, the permutations for the center stage of a Clos network that result in the realization of that permutation can be identified by *m-edge-colorings* of its corresponding bipartite multigraph. A bipartite multigraph is said to be *m-edge-colored* if its edges are colored such that no two edges of the same color are incident on the same vertex and exactly m colors are used. It is well known that m colors suffice to edge-color a bipartite multigraph of degree m [11].

Furthermore, it is well known that there is a correspondence between an m-edge-coloring of a bipartite multigraph representing a permutation π for a Clos network, and the center-stage permutation π_M. (See [21], [24] for an early application and [7] for a thorough treatment.) An example is depicted in Fig. 2 for a Clos network with $m = 2$, $k = 4$. Edges between left and right vertices that are labeled with color 0 are assigned to the upper switch in the center stage, and those colored with 1 are assigned to the lower switch.

In order to exhibit the dependencies of the permutations of the center-stage switches on the permutations to be realized by a Clos network we need to consider m-edge-colorings of the bipartite multigraph representations of certain permutations. These permutations will be identified by some particular subgraphs which we shall refer to as *balanced multiloops.* We will show that these subgraphs can be edge-colored only two different ways. This fact will then later be used in Section IV in the construction of dependencies between the inputs and outputs of a router for three-stage Clos networks.

Definition 4: A bipartite multigraph, (I, M, O), is a *multiloop* of size l if $|I| = l = |O|$ and, if, when multiple edges between pairs of vertices are taken as one, the graph consists of a single loop. □

[3] A multiset is a set that may contain multiple copies of a single element, for example, $\{a, a, b, c\}$.

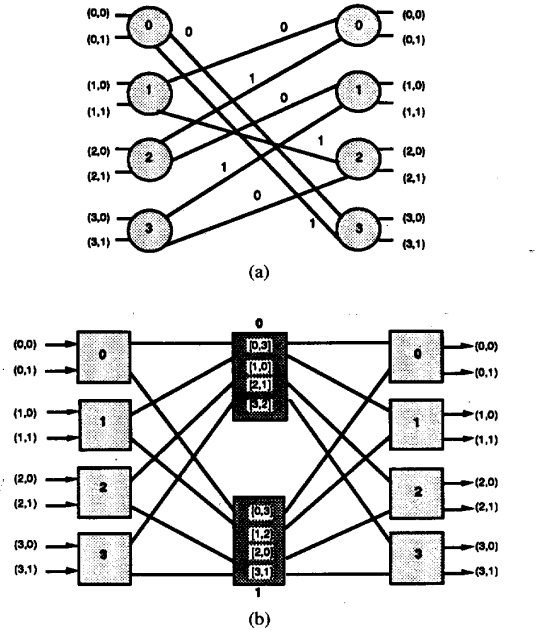

Fig. 2. A bicolored bipartite multigraph and corresponding Clos network.

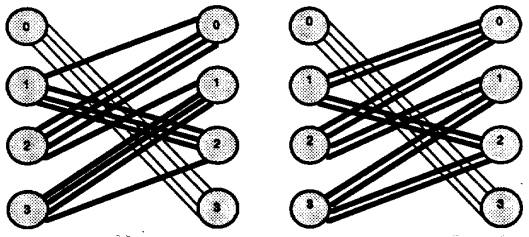

Fig. 3. Examples of multiloops shown in heavy lines for $m = 2$ and $k = 4$.

Definition 5: A bipartite multigraph, (I, M, O), *contains* a multiloop of size l if there exists $I_L \subseteq I$, $O_L \subseteq O$, and $M_L = \{[x, y] \mid x \in I_L, y \in O_L, [x, y] \in M\}$ such that $|I_L| = l = |O_L|$ and the graph (I_L, M_L, O_L) is a multiloop. □

Definition 6: A multiloop (I_L, M_L, O_L) is called *balanced* if it is regular with degree m and for all pairs $[i, j]$, $i \in I_L$ and $j \in O_L$, M_L contains either no edges or exactly $m/2$ edges between vertex i and vertex j. That is to say, every pair of vertices which are connected are connected by $m/2$ edges. □

Examples of ordinary (i.e., not balanced) and balanced multiloops are given in Fig. 3 for a Clos network with $m = k = 4$ where the heavy lines represent the edges in the loops. In both loops, $I_L = \{1, 2, 3\}$, $O_L = \{0, 1, 2\}$.

When a bipartite multigraph is m-edge-colored, it induces an m-edge-coloring on any of its multiloops. For example, when the graphs in Fig. 3 are colored, each of the multiloops ends up having an edge-coloring with four colors. In general, a multiloop of a bipartite multigraph of degree m can be edge-colored by using m colors, and no less since the degree of each vertex in any such loop is m. There exists a bijective mapping between these m colors and the m center-stage switches of the corresponding Clos network in that all the edges in a multiloop with the same color are mapped to the same center-stage switches. Obviously, this mapping is not unique, as the colors can be mapped to the switches in any one of $m!$ ways. In the case

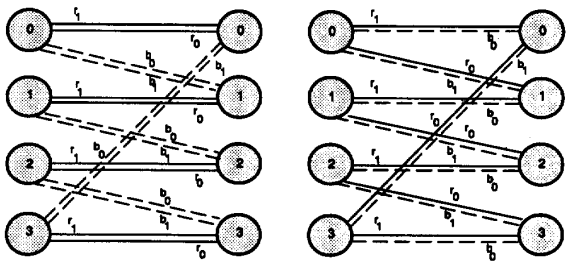

Fig. 4. Examples of unmixed and mixed colorings of a multiloop.

of balanced multiloops, however, the number of such maps can be reduced to two cases as argued below.

Definition 7: Divide the set of m colors assigned to the edges of a multiloop into two sets of colors; *shades of blue,* SB $= \{b_0, b_1, \ldots, b_{m/2-1}\}$, and *shades of red,* SR $= \{r_0, r_1, \ldots, r_{m/2-1}\}$. An m edge-coloring of a multiloop is said to be *unmixed* if the edges between any two vertices receive colors from SR or from SB but not both. Otherwise, it is called mixed. □

Examples of unmixed and mixed edge-colorings of a multiloop are shown in Fig. 4. It can be seen that in the unmixed coloring, the edges (whenever they exist) between a left vertex and right vertex are colored either with shades of red, i.e., r_0 and r_1, or shades of blue, i.e., b_0, and b_1. On the other hand, the mixed coloring does not follow this convention as can be seen in Fig. 4(b).

Lemma 3: A balanced multiloop can have exactly two unmixed m edge-colorings when edges between a pair of vertices are not considered distinct. □

IV. Construction of Dependencies

We proceed by constructing two permutations based on two balanced multiloops.

Definition 8: Let there be two sets of four vertices $\{u_0, u_1, u_2, u_3\} \subseteq I$, and $\{v_0, v_1, v_2, v_3\} \subseteq O$ where I and O are two k-sets of vertices. We construct two multiloops $(I_{\text{loop}}, M_{\text{loop}}, O_{\text{loop}})$ and $(I_{\text{loop}}, M'_{\text{loop}}, O_{\text{loop}})$ such that

$$I_{\text{loop}} = \{u_0, u_1, u_2, u_3\} \subseteq I,$$

$$O_{\text{loop}} = \{v_0, v_1, v_2, v_3\} \subseteq O,$$

$$M_{\text{loop}} = \{[u_0, v_0]^{m/2}, [u_1, v_1]^{m/2}, [u_2, v_2]^{m/2}, [u_3, v_3]^{m/2}, \times [u_0, v_1]^{m/2}, [u_1, v_2]^{m/2}, [u_2, v_3]^{m/2}, [u_3, v_0]^{m/2}\} \subset (I \times O)^{m/2}$$

and,

$$M'_{\text{loop}} = \{[u_0, v_0]^{m/2}, [u_0, v_1]^{m/2}, [u_1, v_0]^{m/2}, [u_1, v_2]^{m/2}, \times [u_2, v_2]^{m/2}, [u_2, v_3]^{m/2}, [u_3, v_1]^{m/2}, [u_3, v_3]^{m/2}\} \subset (I \times O)^{m/2}$$

where $[u, v]^{m/2}$ denotes "edge $[u, v]$ repeated $m/2$ times." A permutation whose bipartite multigraph (I, M, O) has $(I_{\text{loop}}, M_{\text{loop}}, O_{\text{loop}})$ as a subgraph is denoted π_{loop}. Similarly, a permutation whose bipartite multigraph (I, M, O) has $(I_{\text{loop}}, M'_{\text{loop}}, O_{\text{loop}})$ as a subgraph is denoted π'_{loop}. The subgraphs of these permutations will be referred to as loops.

Obviously this extension is not unique, and many permutations can be constructed this way. The two permutations are similar in two respects. Their bipartite multigraph representations both have

TABLE I

Permutation	Coloring	$\pi_M(c, u_0)$	$\pi_M(c, u_2)$
π_{loop}	r	(c, v_0)	(c, v_2)
π_{loop}	b	(c, v_1)	(c, v_3)
π'_{loop}	r	(c, v_0)	(c, v_3)
π'_{loop}	b	(c, v_1)	(c, v_2)

the same left and right vertices and most edges in common: The only difference is that M_{loop} contains edges $[u_1, v_1]^{m/2}$ and $[u_3, v_0]^{m/2}$ which M'_{loop} does not and M'_{loop} contains edges $[u_3, v_1]^{m/2}$ and $[u_1, v_0]^{m/2}$ which M_{loop} does not. Finally, the two permutations contain balanced multiloops of size 4.

By Lemma 3 each of these loops can be edge-colored using shades of two colors in exactly two ways. Table I lists the correct mappings for inputs u_0 and u_2 of a center-stage switch c under π_M that can route π_{loop} and π'_{loop}. Here, r indicates that the edges between vertices u_0 and v_0 are colored with colors from SR and b indicates that these same edges are colored with colors from SB. $\pi_M(c, u_0)$ denotes the output of switch c to which π_M maps input u_0 of that switch. $\pi_M(c, u_2)$ is similarly defined. We also emphasize that output c of input stage switch u_x connects to input u_x of center-stage switch c; similarly, output v_y of center-stage switch c connects to input c of output-stage switch v_y for all $x, y \in \mathbf{Z}_k$. (See Table I.)

Suppose $\Delta = \{(c, u_0), (c, u_2)\}$. Then the possible mappings for the inputs in Δ under π_{loop} and π'_{loop} are $V_\Delta(\pi_{\text{loop}}, \pi_M) = \{[(c, v_0), (c, v_2)], [(c, v_1), (c, v_3)]\}$, and $V_\Delta(\pi'_{\text{loop}}, \pi_M) = \{[(c, v_0), (c, v_3)], [(c, v_1), (c, v_2)]\}$. Let $\mathcal{U}_{u_i} = \{(u_i, j) \mid j \in \mathbf{Z}_m\}$ be the set of inputs of input-stage switch u_i. Since $V_\Delta(\pi_{\text{loop}}, \pi_M) \cap V_\Delta(\pi'_{\text{loop}}, \pi_M) = \phi$, and π_{loop} and π'_{loop} differ only on assignments in sets $\mathcal{U}_{u_1}$ and $\mathcal{U}_{u_3}$, by Definition 3, we conclude that mappings for the inputs in Δ must depend on assignments in $\mathcal{U}_{u_1} \cup \mathcal{U}_{u_3}$.

Theorem 1: For all distinct $u_0, u_1, u_2, u_3 \in \mathbf{Z}_k$ and $c \in \mathbf{Z}_m$, mappings for inputs (c, u_0), (c, u_2) of the center-stage switch c depend upon assignments for inputs in $\mathcal{U}_{u_1} \cup \mathcal{U}_{u_3}$.

Now, consider a family of loop permutations π_{loop} and π'_{loop} in which u_0 and u_2 are the same in each permutation, but where u_1 and u_3 can take on arbitrary values. Regardless of how u_1 and u_3 are chosen, $V_\Delta(\pi_{\text{loop}}, \pi_M) \cap V_\Delta(\pi'_{\text{loop}}, \pi_M) = \phi$, whence

Theorem 2: For all distinct $u_0, u_2 \in \mathbf{Z}_k$ and $c \in \mathbf{Z}_m$, the mappings for inputs (c, u_0), (c, u_2) depend upon the assignments in each of at least $k - 3$ of $\mathcal{U}_i$, $i \in \mathbf{Z}_k \backslash \{u_0, u_2\}$.

Proof: For $u_0, u_2 \in \mathbf{Z}_k$, choose $u_1, u_3 \in \mathbf{Z}_k \backslash \{u_0, u_2\}$. Construct π_{loop} and π'_{loop} from these values. By Lemma 2 and Theorem 1 the $k - 3$ dependencies are obtained. □

Theorem 3: For all distinct $u_0, u_2 \in \mathbf{Z}_k$ and $c \in \mathbf{Z}_m$, mappings for inputs (c, u_0), (c, u_2) in π_M depend upon at least $(k-3)(m/2+1)$ assignments.

Proof: If $\Delta = \{(c, u_0), (c, u_2)\}$ depends on assignments on $\mathcal{U}_i$, then it must depend on assignments on every subset of $m/2$ elements in $\mathcal{U}_i$ since, for every $m/2$ elements chosen out of $\mathcal{U}_i$, we can define two permutations (by appropriate choice of SR and SB) which conform to the structure of π_{loop} and π'_{loop}. Therefore, by Lemma 1, there are at most $m/2 - 1$ assignments Δ does not depend upon. Combining this fact with Theorem 2 we conclude that there are $(k - 3)(m/2 + 1)$ assignments which Δ depends upon. □

Note that Δ contains two elements. Since it cannot be determined which of the two elements depend upon which assignments, the assignments must be divided evenly, to have a correct (worst case) lower bound on the number of dependencies. Thus, a setting in π_M depends upon at least $(k - 3)(m/2 + 1)/2 = (k - 3)(m + 2)/4$ assignments.

Although the dependencies were derived for the settings of the center stage switches, they apply to all stages. That is, permutations π_{loop} and π'_{loop} induce disjoint settings in input-stage switches u_0 and u_2 and output-stage switches v_2 and v_3, so that the same dependencies apply.

The result just derived does not hold for Clos networks with $k \leq 3$. This is because we cannot form balanced loops when $k \leq 3$. To find dependencies in these networks consider a router, this time, for the input-stage switches, and consider the mapping of input (x, y), i.e., input y of switch x in the input stage. This result will be derived for a Clos network in which switch 0 in the input stage is fixed in the identity state. It has been shown that such networks can realize all permutation connections [31].

Theorem 4: The mapping of input (x, y) depends upon at least $m - 1$ assignments.

Proof: Construct two permutations π and π' in the following way:

$$\pi(0, i) = \begin{cases} (0, i) & 0 \leq i < z \\ (1, 0) & i = z \\ (0, i) & z < i < m \end{cases}$$

$$\pi(x, y) = (0, z),$$

$$\pi'(0, i) = \begin{cases} (0, i) & 0 \leq i < w \\ (1, 0) & i = w \\ (0, i) & w < i < z \\ (0, w) & i = z \\ (0, i) & z < i < m \end{cases}$$

$$\pi'(x, y) = (0, z)$$

where $w, y, z \in \mathbf{Z}_m$, such that $w \neq z$, and $x \neq 0$, and where unspecified, assignments are identical in π and π'. Permutations π and π' differ only in inputs $(0, w)$ and $(0, z)$. With permutation π, the only path from input-stage switch x to output-stage switch 0 is through center-stage switch z. With permutation π' the only path from input-stage switch x to output-stage switch 0 is through center-stage switch w. Therefore, the mapping of input (x, y) in π_I must be different under permutations π and π', and consequently, depends on assignments for inputs $(0, w)$ and $(0, z)$. But, since z and w can take on all unequal pairs of values, by Lemma 1 the mapping of input (x, y) depends upon at least $m - 2 + 1 = m - 1$ inputs. □

Combining Theorem 3 and Theorem 4, we conclude the following.

Corollary 1: The information complexity of routing an n-input Clos network is $(k-2)(n/k+2)/4$, for all $k \geq 4$, and it is $(n/k)-1$ for $k \leq 3$. Furthermore, if the fan-in of the devices used in a router to route an n-input Clos network is no more than w then it takes $\Omega(\log_w n)$ time to compute the switch settings in that network. □

V. Routing Complexity of Recursively Decomposed Clos Networks

With the lower time bounds established in the preceding section, we can now bound the time of routing a recursively decomposed Clos network. This bound depends on how the switches in the center stage are routed as well as on how much time we allocate to routing the remainder of the network.

In a recursively decomposed Clos network, some switches are implemented as smaller Clos subnetworks. Particularly, switches below a certain size are directly implemented using crossbar switches. Larger switches are realized by smaller Clos networks, called subnetworks in this context. For example, a 2^s-input Beneš network is a recursively decomposed Clos network with $m = 2$; the value of k varies with the level of recursion. If $k = 2^{l-1}$ then $l = s$ at the

TABLE III

Routing Algorithm	Network Types	Level Time	Total Time	Maximum Time Condition
Lower Bound	$2 \leq m \leq n$	$\log_w n$	$\Omega(\log_m n \log_w n)$	$m = 2$
Nassimi and Sahni	$m = 2$	$\log n$	$\Theta(\log^2 n)$	—
Lev *et al.*	m, power of 2	$\log m \log n$	$\Theta(\log^2 n + \log m \log(n/m^2))$	$m = n^{0.25}$
Lev *et al.*	$2 \leq m \leq n$	$\log m \log^2 n$	$\Theta(\log^2 n \log n^2 m^3 + \log^2 m \log(n/m^6))$	$m = n^{((1/18)+\sqrt{55/324})}$

TABLE II

Routing Scheme	Recurrence Relation	Routing Time Lower Bound
R-sequential/L-sequential	$T_n = mT_{n/m} + cn$	$\Omega(n \log_m n)$
R-parallel/L-sequential	$T_n = T_{n/m} + cn$	$\Omega(n)$
R-sequential/L-parallel	$T_n = mT_{n/m} + c \log_w n$	$\Omega(n + (n/m) \log_w n)$
R-parallel/L-parallel	$T_n = T_{n/m} + c \log_w n$	$\Omega(\log_m n \log_w n)$

top level, $l = s - 1$ at the second level of recursion, and so on. The input and output stage switches are directly implemented using crossbar switches, while each of the two center stage switches are implemented as 2^{l-1}-input Beneš networks at level l.

Routers for recursively decomposed networks can be specified in a recursive fashion: the input to the router is the permutation the network is to realize; the router generates the setting for each of the switches, and is invoked recursively for the settings of the subnetworks. To route the subnetworks, i.e., to find their settings, we can proceed in at least one of two ways: either set all the subnetworks in the center in parallel, or set them one at a time in some predetermined sequential order. We will refer to these cases as *R-parallel* and *R-sequential* schemes in that order. It is possible to use other routing schemes, e.g., set the first half of subnetworks in parallel, and then set the second half in parallel, etc. Even though these intermediate schemes may provide further insight on routing recursively decomposed Clos networks, here we are interested in bounds on routing complexity, and therefore will restrict our focus to R-parallel and R-sequential schemes.

As for routing each level, i.e., deciding the settings of the switches at each level, we may proceed in two different ways. Any sequential routing scheme can be seen to require $\Omega(n)$ time (reading in the permutation to be realized will take $\Omega(n)$ time) while the lower bound of the previous section indicates that any parallel scheme on a router with fan-in w takes $\Omega(\log_w n)$ time. Routing schemes which confirm to these lower bounds will, respectively, be referred to as *L-parallel* and *L-sequential* schemes.

Table II lists the four routing schemes which are formed by combining the recursion and level choices. The first column lists the possible choices, the entries in the second column are the recurrence relations corresponding to these choices, and those in the third column are execution time complexities for the four routing schemes. The variable c in the recurrences is a constant. The derivations of these expressions from the recurrences are straightforward and omitted here.

It is seen that any R-sequential/L-sequential routing scheme requires $\Omega(n \log_m n)$ time. When $m = 2$ this reduces to $\Omega(n \log n)$, and when $m = n/2$ it reduces to $\Omega(n)$. The first case characterizes networks with small switches in their outer stages such as the Beneš network [5], while the second case characterizes networks with large switches in their outer stages such as the complementary Beneš network [7]. On the other hand, any R-parallel/L-sequential scheme requires $\Omega(n)$ time, for all integral m, $2 \leq m \leq n$.

As for the last two routing schemes, it follows from the table that any R-sequential/L-parallel scheme takes $\Omega(n)$ time when $m = O(n)$ and $\Omega(n \log_w n)$ time when $m = O(1)$. Furthermore, any R-parallel/L-parallel scheme needs $\Omega(\log_m n \log_w n)$ time which reduces to $\Omega(\log_w^2 n)$ when $w = \Theta(m)$.

VI. Comparisons

In this section the lower bounds on the routing time of Clos networks will be compared to the routing times of existing algorithms with and without recursive decomposition. The time complexities of the existing routing algorithms match the lower bounds for the cases of $m = 2$ and $k = 2$, but for intermediate values of m and k, the time complexity of known algorithms is higher. This discrepancy is not surprising, as will be explained below, but also hints of the existence of faster algorithms.

First, consider routing three-stage Clos networks without recursive decomposition. For $m = 2$, the fastest algorithm has $\Theta(\log n)$ time complexity [24], matching the lower bound given in Corollary 1 with $w = O(1)$. For larger m there are algorithms with time complexity $\Theta(\log m \log n)$ when m is a power of 2 and $\Theta(\log m \log^2 n)$ otherwise [21]. Here, the known algorithms take more time than their $m = 2$ and $k = 2$ counterparts, while the bound is lower. The reason that the known algorithms take longer when $m > 2$ is because they treat the Clos network as a Beneš network; they are iterated $\Theta(\log m)$ times, each subsequent call routes a lower level in R-parallel fashion.

When $m > 2$, a faster algorithm might avoid iteration even though it at first seems that there is an as yet undiscovered reason why this cannot be done. However, the existence of a bound-matching routing procedure for Clos networks with $m = n/2$ suggests otherwise. A Clos network with $m = n/2$ can be routed in $\Theta(SO(n/2))$ time where $SO(n/2)$ is the execution time of an $n/2$ item sorting algorithm [19]. Using the AKS sorting algorithm [1] one can then set up the network in $\Theta(\log n)$ time, which matches the bound in Corollary 1 and suggests that iteration is not necessary.

These observations can be extended to recursively decomposed Clos networks. Table III shows the lower bound for any R-parallel/L-parallel routing scheme for a Clos network, and the time complexities of the best known routing algorithms in this case. The parameter m is of particular interest because network cost increases with increasing m while the number of switches between an input and output decreases with increasing m. Because of m's importance the table includes the value of m in the interval $2 \leq m \leq n$ which would yield the maximum routing time. The complexities and maxima were obtained from solutions of the recurrence

$$T_n = T_{n/m} + L(n)$$

where $L(n)$ is the expression in the table's "Level Time" column.

If a level of a Clos network could be routed in $\Theta(\log n)$ time, the entire network could be routed in $\Theta(\log_m n \log_w n)$ time using an

L-parallel/R-parallel algorithm. This bound is achieved with $m = 2$ and $w = O(1)$ using the algorithm of Nassimi and Sahni [24] and Lev *et al.* [21]. With larger m, the bound on time drops; this occurs both at a single level and for the entire network; the routing time bound is at a maximum when $m = 2$. As with the single level case, the routing time of the known algorithms for recursively decomposed Clos networks is greater than the bound by a logarithmic factor when $m = \Theta(n)$ and is a power of two and by a log squared factor otherwise.

Consider the Lev *et al.*'s algorithm when m is a power of two. As can be verified from Table III, the time complexity of this algorithm is $\Theta(\log^2 n)$ for any m which is a power of 2, while the lower bound does not rule out the possibility of a $\Theta(\log n)$-time algorithm. As stated above, the only value of m for which this lower bound is attainable is $n/2$. This is achieved by using the sorting-based routing algorithm [19] with its sorting steps realized by the AKS sorting network. Given that the AKS sorting network remains impractical, a Batcher odd–even merge or bitonic sorter [4] can be used to implement the sorting steps of this algorithm, in which case a routing time of $\Theta(\log^2 n)$ can be achieved as with Lev *et al.*'s algorithm.

Finally, Lev *et al.*'s algorithm exhibits an even higher order of routing time complexity when $2 \leq m \leq n$ as manifested by the expression in the last row. A faster Clos network routing algorithm for intermediate values of m should prove useful. Although recursively decomposed Clos networks for $m = 2$ have fewer crosspoints [5] the cost may be smaller for other values of m, when additional factors such as the overhead of implementing a cell of any size, the cost of interstage links, etc., are taken into account. Thus, there is a need for fast routing algorithms for Clos networks with other values of m.

VII. Concluding Remarks

The paper has presented lower bounds on the routing time of Clos networks. It has been shown that, when $k > 3$, at least $(k-3)(m/2+1)/2$ assignments in the permutation to be routed must be examined in order to compute the switch settings of a Clos network with k input-stage switches, each encompassing m inputs. It has additionally been shown that, when $k \leq 3$, at least $m-1$ assignments in the permutation to be routed must be examined in order to compute the switch settings of the same network. Combining these results with a simple fan-in argument then gives $\Omega(\log_w n)$ bound on the routing time of a three-stage Clos network on any w fan-in limited routing model. Implications of this bound were considered for recursively decomposed Clos networks with respect to four routing schemes.

Acknowledgment

The authors thank all three reviewers for their careful reading of the paper and constructive suggestions.

References

[1] M. Ajtai, J. Komlós, and E. Szemeredi, "Sorting in $c \log n$ steps," *Combinatorica*, vol. 3, pp. 1–19, 1983.
[2] S. Andresen, "The looping algorithm extended to base 2^t rearrangeable switching networks," *IEEE Trans. Commun.*, vol. COM-20, pp. 1057–1063, 1977.
[3] L. A. Bassalygo and V. I. Neiman, "On two control algorithms for rearrangeable switching networks," in *Proc. 10th Int. Teletraff. Congr.*, Monterey, CA, 1983, pp. 5.1:7/1–5.1:7/7.
[4] K. E. Batcher, "Sorting networks and their applications," *AFIPS, Spring Joint Comp. Conf.*, 1968, pp. 307–314.
[5] V. E. Beneš, "Optimal rearrangeable multistage connecting networks," *Bell Syst. Tech. J.*, vol. 43, no. 4, pp. 1641–1656, July 1964.
[6] G. Broomell and J. R. Heath, "Classification categories and historical development of circuit switching topologies," *ACM Comput. Surv.*, vol. 15, no. 2, pp. 95–133, June 1983.
[7] J. D. Carpinelli and A. Y. Oruç, "Applications of edge coloring algorithms to routing in parallel computers," in *Proc. 3rd Int. Conf. Supercomput.*, Boston, MA, 1988, vol. III, pp. 249–257.
[8] C. Clos, "A study of non-blocking switching networks," *Bell Syst. Tech. J.*, vol. 32, no. 2, pp. 406–424, Mar. 1953.
[9] R. Cole and J. Hopcroft, "On edge-coloring bipartite graphs," *SIAM J. Comput.*, vol. 11, no. 3, pp. 540–546, Aug. 1982.
[10] T.-y. Feng, "A survey of interconnection networks," *IEEE Comput.*, pp. 12–27, Dec. 1981.
[11] S. Florini and R. J. Wilson, *Edge-Coloring of Graphs.* London, England: Pitman, 1977.
[12] H. N. Gabow, "Using Euler partitions to edge color bipartite multigraphs," *Int. J. Comput. Inform. Syst.*, vol. 5, no. 4, pp. 345–355, 1976.
[13] H. Gabow and O. Kariv, "Algorithms for edge coloring bipartite graphs and multigraphs," *SIAM J. Comput.*, vol. 11, no. 1, pp. 117–129, Feb. 1982.
[14] F. K. Hwang, "Control algorithms for rearrangeable Clos networks," *IEEE Trans. Commun.*, vol. COM-31, pp. 952–954, Aug. 1983.
[15] A. Jajszczyk, "A simple algorithm for the control of rearrangeable switching networks," *IEEE Trans. Commun.*, vol. COM-33, pp. 169–171, Feb. 1985.
[16] R. M. Karp and V. Ramachandran, "Parallel algorithms for shared-memory machines," in *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity.* New York: Elsevier, 1990, pp. 869–941.
[17] D. E. Knuth, "Big omicron and big omega and big theta," *SIGACT News*, vol. 8, pp. 18–24, Apr. 1976.
[18] D. M. Koppelman and A. Y. Oruç, "Parallel time complexity of routing in permutation networks," in *Twenty-Sixth Annu. Allerton Conf. Commun., Contr., Comput.*, Sept. 1988, pp. 981–990.
[19] ——, "A self-routing permutation network," *J. Parallel Distrib. Computing*, vol. 10, pp. 140–151, 1990.
[20] C. P. Kruskal and M. Snir, "A unified theory of interconnection network structures," *Theoretical Comput. Sci.*, vol. 48, pp. 75–94, 1986.
[21] G. F. Lev, N. Pippenger, and L. G. Valiant, "A fast parallel algorithm for routing in permutation networks," *IEEE Trans. Comput.*, vol. C-30, pp. 93–100, Feb. 1981.
[22] M. J. Marcus, "The theory of of connecting networks and their complexity: A review," *Proc. IEEE*, vol. 65, pp. 1263–1271, Sept. 1977.
[23] G. M. Masson, G. C. Gingher, and S. Nakamura, "A sampler of circuit switching networks," *IEEE Comput.*, vol. 12, pp. 32–48, June 1979.
[24] D. Nassimi and S. Sahni, "Parallel algorithms to set up the Beneš permutation networks," *IEEE Trans. Comput.*, vol. C-31, pp. 148–154, Feb. 1982.
[25] D. Nassimi and S. Sahni, "A self-routing Beneš network and parallel permutation algorithms," *IEEE Trans. Comput.*, vol. C-30, pp. 332–340, May 1981.
[26] V. I. Neiman, "Structure et command optimales de reseaux de connexion sans blocage," *Ann. Telecommun.*, vol. 24, pp. 232–238, July–Aug. 1969.
[27] D. C. Opferman and N. T. Tsao-Wu, "On a class of rearrangeable switching networks, part I: Control algorithms, part II: Enumeration studies and fault diagnosis," *Bell Syst. Tech. J.*, vol. 50, no. 5, pp. 1579–1618, May 1971.
[28] N. Pippenger, "Telephone switching networks," in *Proc. Symp. Appl. Mathemat.*, vol. 26, pp. 101–113, May 1982.
[29] C. S. Raghavendra and R. V. Boppana, "On self-routing in Beneš and shuffle-exchange networks," *IEEE Trans. Comput.*, vol. 40, pp. 1057–1065, Sept. 1991.
[30] C. E. Shannon, "Memory requirements in a telephone exchange," *Bell Syst. Tech. J.*, vol. 29, pp. 343–349, 1950.
[31] A. Waksman, "A permutation networks," *J. Assoc. Comput. Machinery*, vol. 15, no. 1, pp. 159–163, Jan. 1968.