A New Evaluation Criterion for Clos- and Benes-Type Rearrangeable Switching Networks

Ioannis Gragopoulos, *Student Member, IEEE,* and Fotini-Niovi Pavlidou, *Member, IEEE*

***Abstract*—An extended comparison and a classification of the control algorithms for rearrangeable switching networks is tried in this study. Besides of the failure probability P_f a new evaluation criterion is introduced, the mean number of rearranges R_m the network performs in order to satisfy incoming calls. A simulation comparison for eight control algorithms concerning clos-type networks and networks with 2×2 elements is attempted with very helpful results for the network design. Furthermore, some deadlock conditions discovered in one of these algorithms are completely recovered. The comparison is carried out for both evaluation criteria (R_m and P_f) and it is shown that the new criterion leads to a deep insight of the systems, necessary for a successful network design.**

***Index Terms*—Rearrangeable networks, routing, switching.**

I. Introduction

In a rearrangeable network we can satisfy an incoming call after properly changing the previously established paths. The most studied structure of rearrangeable networks is the Clos network, but 2×2 structures have also been proposed recently for high speed networks. In this study we examine both cases: Clos structures for analog networks and Benes structures for modern high speed telecommunications networks.

A. Clos-Type Networks

A $N \times N$ Clos-type network [7], is a three-stage switch in which the N inlets (outlets) are partitioned into m subgroups of n inlets (outlets) and the center-stage consists of n subgroups with m inlets and outlets each. For rearrangeability the condition for zero failure probability is $n \geq m$.

Various control algorithms for Clos-type switching networks have been extensively studied the last decades, in order to decrease the size of the switch, assuming in parallel a high capability to realize all possible permutations. These control algorithms can be developed generally after two different techniques. In the first technique connections can be established on a call-by-call basis, i.e., a connection is set up for each new call, rearranging if necessary some of the existing connections. This is the well-known, classical Paul Matrix technique and its modifications [7]. In the second technique, a parallel processing of the incoming calls is carried out, that is, connections are studied in batch for all input-output pairs. Parallel control algorithms are further splitted in matrix decomposition and scheduling techniques [3]–[5], [8]. Each one of them is showing advantages and disadvantages, if we examine its running time of computation, or the failure probability. Recently, an algorithm [5], [4] has been proposed which presents a really improved performance over the others. In this paper an in-depth study of the characteristics and the possibilities of all these algorithms is tried.

Paper approved by D. Kazakos, the Editor for Random Access and Distributed Communications Systems of the IEEE Communications Society.

The authors are with the Aristotle University of Thessaloniki, School of Engineering, Department of Electrical and Computer Engineering, Division of Telecommunications, Thessaloniki, 54006 Greece (email: grag@amphipolis.ee.auth.gr and niovi@vergina.eng.auth.gr).

Publisher Item Identifier S 0090-6778(97)00724-1.

B. Benes-Type Networks

There are some well established techniques to construct a rearrangeable network starting of a single square matrix. By the recursive procedure it is possible to start with a single square matrix, then substitute it by a three stage network (Clos-type networks) then substitute the matrixes in the center stage, with new three stage networks, etc. For the special case of $N = 2^n$, n being a positive integer, we can recursively construct a rearrangeable network by factoring N into $N/2$ first-stage 2×2 switches and two $N/2 \times N/2$ middle-stage switches. The resulting network is called Benes network [7]. The interconnection networks, with 2×2 elements, studied in this paper are the Benes, the perfect-shuffle and the flip network [7]. One method of establishing the connections in these type networks is via the looping algorithm [7]. The looping algorithm is the most representative serial processing algorithm. Since then more sophisticated control algorithms have been developed. For purposes of comparison four control algorithms have been used: a) An extension of the looping algorithm, established by Andresen [1], b) a generalization of the looping algorithm developed by Lea [9], c) a parallel processing algorithm realized by Huang and Tripathi [6] based on a model that they developed, known as finite permutation machine (FPM), to describe the permutation networks and d) a control algorithm developed by Raghavendra and Varma [10] that is specific only to the case of five-stage shuffle/exchange network for $N = 8$.

We carry out the study and the comparison of control algorithms (for Clos and Benes-type networks and networks with 2×2 elements), applying two evaluation criteria, the mean number of rearranges R_m needed to satisfy changes in the call pattern and the probability of failure P_f, which indicates the capability of the network to service a call. The first criterion is a new criterion and has been proved very valuable for evaluating control algorithms.

Also, after the detection of some fault conditions in the recently proposed algorithm of Chiu–Siu [4] we introduce a simple but effective correction technique.

The rest of the paper is organized as follows: In Section II we present the existing algorithms for clos-type networks, in Section III we present shortly the existing algorithms for the case of rearrangeable networks with 2×2 elements, in Section IV we give numerical comparative results and in Section V the conclusions. In Appendix A we give an example where the Chiu–Siu algorithm fails and we present our correction technique. In Appendix B a comparison of the control algorithms for Clos type networks is given, using the failure probability P_f.

II. Description of Algorithms for Clos-Type Networks

Three algorithms are the most representative for parallel rearrangeable control. Jajszczyk algorithm and Carpinelli–Oru algorithm based on matrix decomposition techniques and Gordon–Srikanthan algorithm based on the scheduling method. Only the Carpinelli–Oru algorithm always succeeds to route any permutations through the Clos network. We give a brief description of the algorithms pointing out details of their performance. The call pattern is given by a permutation vector P of input–output connections in any of these algorithms.

A. Jajszczyk Algorithm

For a given permutation P, a $m \times m$ square matrix $H_m^{(n)}$ is constructed, where each element $[h_{ij}]$ denotes the number of connections from the ith first-stage switch to the jth third-stage switch $(1 \leq i \leq m, 1 \leq j \leq m)$ [8]. An elementary permutation matrix E is one for which the sum of any row and column is unity. The objective purpose of the algorithm is to extract from the matrix H, n square matrixes E of $m \times m$ dimension each. The E matrixes denote permutations realized by the middle-stage switches for path establishment. The computation complexity of the algorithm is $O(nm^2)$.

The problem is that in many cases matrix H results in rows or columns with only zero elements leading the algorithm to deadlocks. Because of the random selection of nonzero elements we cannot predefine what the result will be after a few repetitions. One way to solve deadlock situations is the backtracking method. With the backtracking method the algorithm goes back in order to select a different nonzero element. We can backtrack as many repetitions as necessary to solve the deadlock, (if there is any solution), or we can backtrack by jumping back a fix number of steps. Both techniques are checked in Appendix B.

B. Carpinelli–Oru Algorithm

The algorithm—based on Jajszczyk method—works on a trial, to predefine and avoid deadlock situations, leading to matrix H partitioning (existence of zero rows or columns) [3].

The algorithm begins by setting up the matrix H_m in the usual way. Then the matrix is checked for possible partitions. If no partitions are found the algorithm proceeds as the Jajszczyk method. If a partition exists, H_m matrix is divided into two submatrixes and the whole procedure is applied for each submatrix in a retrospective way. As it is obvious the algorithm never fails so it needs no backtracking. But all the possible subsets of rows of the matrix H_m must be found and cross checked with all the columns of H_m. This procedure is time consuming and leads to a sophisticated code. The complexity of the algorithm is $O(2^k)$ using one processor and $O(k)$ using 2^k processors, where 2^k are all the possible sets of rows of the matrix H_m.

C. Gordon–Srikanthan Algorithm

Gordon and Srikanthan [5] developed a control algorithm based on a completely new technique called scheduling. One year later Chiu and Siu proved that Gordon–Srikanthan algorithm falls to deadlocks for some permutations and they presented a modification [4] of the original algorithm.

The core of this algorithm consists of two digital matrixes S and C with ones and zeroes. For a given permutation P the $m \times n$ matrix S is constructed, where each element $[s_{ij}]$ denotes that the jth input of the ith first-stage switch, demands connection with the s_{ij} third-stage switch $(1 \leq i \leq m, 1 \leq j \leq n)$. Then the $m \times n$ matrix C is constructed where each element $[c_{ij}]$ denotes the number of occurrences of the number i in the column j of the S matrix. The algorithm continues by swapping the elements of S until the matrix C contains only ones. The solution is obtained by observing that after the manipulation of the S matrix, the s_{ij} element may be interpreted as the output subnetwork to which the signal from input stage i and center stage j is routed.

Even if the Chiu–Siu algorithm is very recent and well performing we have found after extended computer simulation applications that the algorithm fails to converge in some cases. We give a brief example and our correction technique in Appendix A.

III. Description of Algorithms for Networks with 2×2 Elements

A. Andresen Algorithm

The extended looping algorithm is based on decomposition technique, generating a suitable decomposition in a pure sequential manner. An arbitrary permutation P is represented by a $N \times N$ dimensions matrix M. A x in the position (i, j) of the matrix M indicates that the input i is to be connected to the output j. The algorithm results to an index in every x in the matrix M indicating the number (in binary form) of the center switches through which the permutation must be routed.

Although in the original paper [1] the algorithm is applied to a Clos network, we introduce an application for the case of networks with 2×2 elements in a recursive manner. The resulting code for an arbitrary permutation P with N inputs–outputs, is presented in the following steps.

1) Set $k = 1$.
2) Set $i = 0$.
3) Construct the permutation set $P_s = \{P_1|, \cdots, |P_k\}$ where each element P_k is a new permutation row vector with N/k elements. The P_k matrixes denote the new permutations that will be established to the k subnetwork in the center stage. For the first repetition $P_s = P$.
4) Set $j = 1$.

5) Apply the extended looping algorithm for each P_j.
6) Set $j = j + 1$. If $j > k$ stop, else goto step 4).
7) Set $k = 2 * k$.
8) Set $i = i + 1$.
9) Compute column i and column $2\log_2 N - i$ of matrix E. The column i of matrix E denotes the switch settings of i stage in binary form.
10) If $i \leq \log_2 N - 1$ go to step 3). Else stop.

B. Finite Permutation Machine (FPM) Algorithm

This algorithm aims to define the settings of 2×2 switches for all Zstages for shuffle-exchange networks. It consists of $3R - 3$ stages with $N/2$ 2×2 switches each, where $R = \log_2 N$. A binary representation of any arbitrary permutation is made and a matrix D with N rows and $\log_2 N$ binary column vectors V_i is constructed, $D = [V_1, V_2, \cdots]$. Inputs are also represented in binary format and a matrix $M_b = [Z_1, Z_2, \cdots]$ is derived, where Z_i are $\log_2 N$ binary column vectors. The algorithm continues by making shuffle and exchange operations in both the D and M_b matrixes and the result of the algorithm is a matrix E in binary form consisting of N rows and $3R - 3$ columns. Each column denotes the settings of the switches of the corresponding stage.

C. Lea Algorithm

The Lea algorithm [9] is actually a generalization of the looping algorithm. Function $p(k)$ converts the inlet vertex k to the number of its butterfly inlet partner. Function $q(k)$ does the same conversion for the outlet vertices. The algorithm begins with the construction of a sequence of outlets and their source inlets. The outlet sequence is denoted by $\{o_{l_1}, o_{l_2}, \cdots\}$ and the inlet sequence is denoted by $\{i_{j_1}, i_{j_2}, \cdots\}$. During the process of composition, the inlets and outlets will be assigned to either the upper or lower $N/2 \times N/2$ subnetwork, and the corresponding states of the switching elements will be determined accordingly. Assuming o_{l_1}, i.e., $l_1 = 1$, assign it to the upper subnetwork and indicate it by the notation $o_{l_1}(U)$. Then o_{l_2}, where $l_2 = q(l)$, has to be assigned to the lower subnetwork, i.e., $o_{l_2}(L)$. Assume the source inlet of o_{l_2} is i_{j_1}, then i_{j_1} has to be connected to the lower $N/2 \times N/2$ subnetwork, i.e., $i_{j_1}(L)$, so that i_{j_1} and o_{l_2} will use the lower subnetwork to establish the connection. This procedure continues until all the inlet-outlet pairs will be assigned to either upper or lower subnetworks without any blocking. The same procedure can be applied to the $N/2 \times N/2$ subnetworks until all the nodes in the network are set up to the correct states.

D. Raghavendra–Varma Algorithm [10]

The algorithm deals exclusively with shuffle-exchange switching networks with maximum number of users $N = 8$, consisting of five stages and four 2×2 switches per stage. For routing arbitrary permutations on the five-stage binary network we split the connections of the permutation into four connection sets, consisting of two connections each, and assign each connection set to one of the switches in the middle stage of the network. This partitioning is done in such a way that no conflicts arise in either the first or the last two stages of the network. The algorithm is executed in six steps two of theme using the well-known looping algorithm.

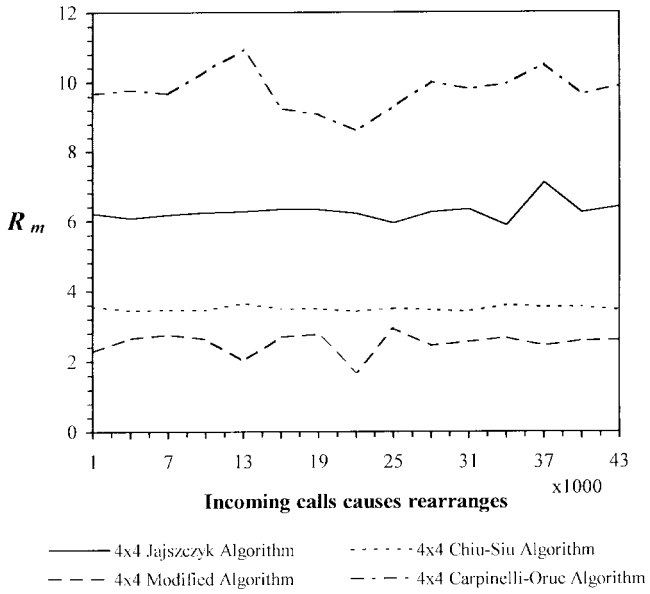


Fig. 1. Mean number of rearranges of the four algorithms for a 4×4 Clos network.

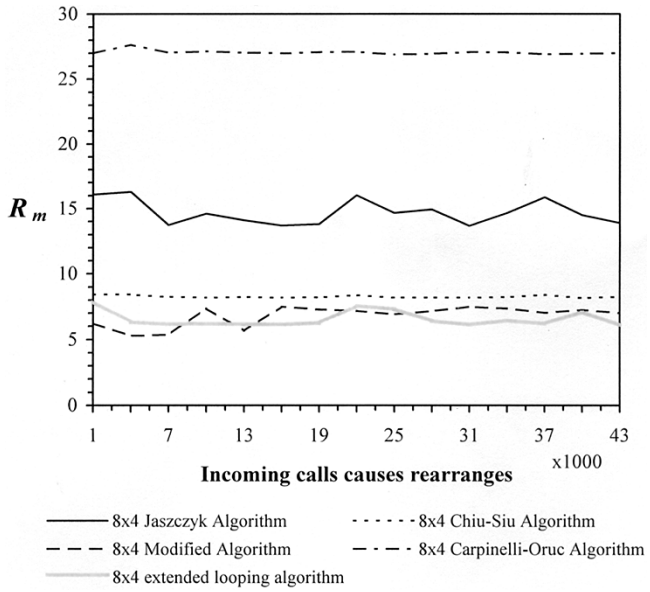


Fig. 2. Mean number of rearranges of the control algorithms (including extended looping algorithm) for a 8×4 Clos network.

IV. Numerical Comparison—The New Performance Criterion

The numerical evaluation of rearrangeable algorithms was always based on the study of the failure probability P_f and the complexity of the codes. As we have mentioned in the introduction we apply for the first time a new criterion, the mean number of rearranges R_m, in order to have an in-depth testing of these algorithms for any network design.

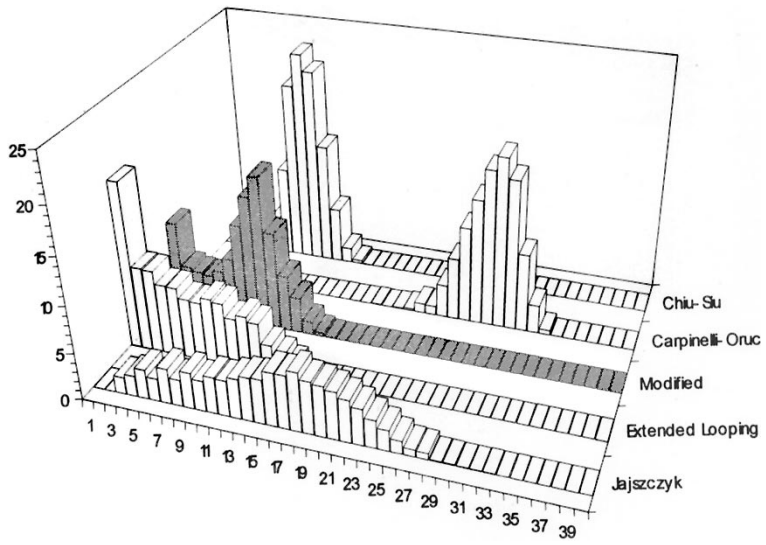


Fig. 3. Histogram of the percentage of permutations for different mean number of rearranges for the control algorithms including the extended looping algorithm.

A. Clos-Type Switching Networks

In the following figures in the X-axis we have the number of new incoming calls causing network rearranges. In Fig. 1 the mean number of rearranges R_m is given for 4×4 Clos network. We observe that our modified algorithm shows the minimum mean number of rearranges and the Carpinelli–Oruc algorithm holds the maximum. The Chiu–Siu algorithm shows the most stable mean value which means that the variance of the number of rearranges is small. A high value of variance declares that the number of rearranges is not easily predictable when altering different elements in the permutation matrix. This result is clearly demonstrated in Fig. 3, where the mean number of rearranges according to the percentage of permutations is given. The Chiu–Siu algorithm holds the minimum variance of the number of rearranges even if the network size increases. The variance of the other algorithms depends on the size of the network. If the network size increases the mean number of rearranges increases too with different percentage for each algorithm as we can see in Fig. 2. We must mention that by our modification we succeed to keep a relatively low variance while getting the lower R_m.

B. Switching Networks with 2×2 Elements

Although the extended looping algorithm can be applied into switching networks with 2×2 elements, it was realized for the clos-type switching networks. So we can try a comparison of it with the previous control algorithms for clos networks. The results are shown in Fig. 2. We can see that the extended looping algorithm presents a mean number of rearranges R_m very close to our modified algorithm. The stability of the algorithm can be seen in Fig. 3, where R_m holds an almost exponential behavior with the percentage of permutations, which is not an acceptable result for network design.

The next investigation concerns control algorithms for networks with 2×2 switching elements and specifically the perfect shuffle network, the benes network and the flip network for various sizes. As it was previously mentioned each control algorithm demands different number of stages in order to realize all the possible permutations. Therefore, a direct comparison of the mean number of rearranges (as in the clos networks) of all control algorithms is of no use, since the number of crosspoints is different for each one of them. Thus, we compare the percent ratio of R_m (R_m%) over the number of crosspoints of each structure.

The first comparison concerns the perfect shuffle network (the Andresen algorithm and the Lea algorithm have been applied for this case). The results for the case of $N = 8$ and $N = 32$ are given in Figs. 4–5 correspondingly. We observe that for small networks ($N = 8$) no clear difference can be recorded in the algorithms performance. The algorithms present unstable performance according to the mean percentage number of rearranges R_m%. As the size of the network increases the performance of the algorithms becomes more stable and for the case of $N = 32$ the Lea algorithm presents almost constant behavior. On the other hand the Huang–Tripathi algorithm has the minimum mean number of rearranges, but with less stable performance.

Results are very similar for Benes and flip networks. More precisely the only significant difference is for the Benes network. This kind of network is constructed by a recursive procedure leading to separate subnetworks inside the switching fabric. In other words, any arbitrary call has less flexibility for allocating the available paths. So, for the Benes network the algorithms present more stable performance as it is shown in Fig. 6. Additionally the mean number of rearranges R_m% for the Andresen algorithm is reduced to a very low value for the case of Benes network.

V. Conclusion

From the above analysis, we can conclude that our modification of Chiu–Siu algorithm presents the most attractive performance in terms of the mean number of rearranges without being very complicated. Specifically it is slightly more complicated than Chiu–Siu algorithm which is really a simple

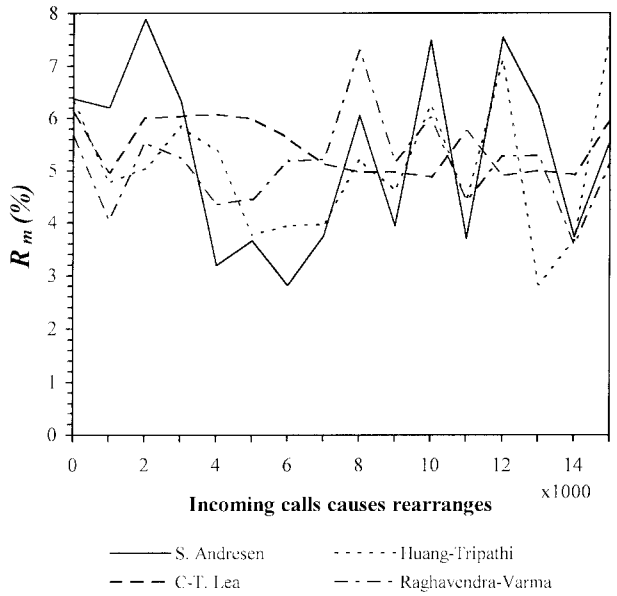


Fig. 4. The four algorithms for the shuffle/exchange network for $N = 8$.

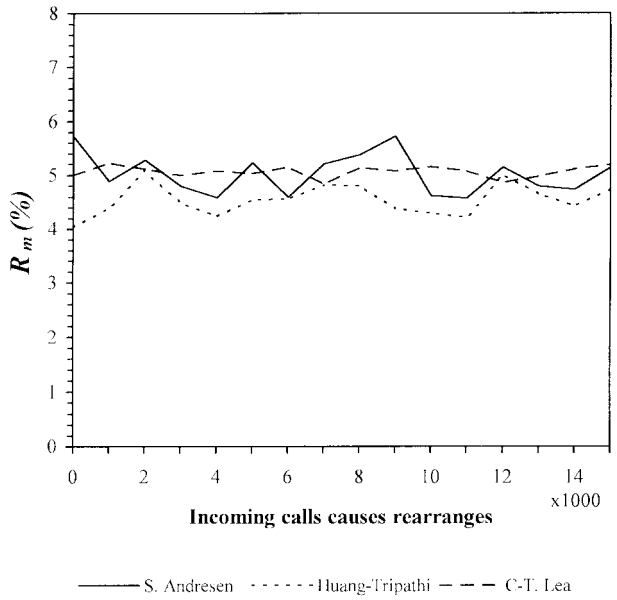


Fig. 5. The four control algorithms for the shuffle/exchange network for $N = 32$.

algorithm. The Andresen algorithm for the Clos network case presents similar behavior to Chiu–Siu modified algorithm. On the other hand the Andresen algorithm is based on the looping algorithm which is a pure sequential algorithm, in contrast to our parallel computing (and more fast in application) modified algorithm. For the case of networks with 2×2 elements the comparison is more complicated. These algorithms are more sensitive to the size of the network in contrast with the control algorithms for Clos networks. In addition, the architecture of the networks plays important role to the mean number of rearranges. We can conclude that Lea algorithm seems to be the more stable according to the size and the architecture of the network, showing a moderate R_m with the lowest variance.

After this analysis, we can state that the new criterion introduced gives us an in-depth knowledge for the properties of each structure, it is applicable both to Clos-type and Benes-type networks and it can give a common base for design of any type of switching network.

Appendix A

Even if the Chiu–Siu algorithm is very recent and well performing we have found after extended computer simulation applications that the algorithm fails to converge in some cases. We give a brief example assuming a 16×16 Clos network with $m = 4$ (first-stage switches) and $n = 4$ (middle-stage switches). We select the following permutation matrix P

$$P = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 11 & 14 & 0 & 12 & 8 & 2 & 6 & 4 & 3 \\ 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 7 & 5 & 9 & 13 & 1 & 15 & 10 \end{bmatrix}.$$

The S and C matrix are given [5] by

$$S = \begin{bmatrix} 2 & 3 & 0 & 3 \\ 2 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 3 & 0 & 3 & 2 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 2 & 0 & 0 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

First repetition: $i = 1$, $j = 0$, $k = 2$, $u = 1$

$$S = \begin{bmatrix} 2 & 3 & 0 & 3 \\ 1^* & 0 & 2^* & 1 \\ 0 & 1 & 1 & 2 \\ 3 & 0 & 3 & 2 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 1^+ & 1 & 1^- & 1 \\ 1^- & 0 & 1^+ & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Second repetition: $i = 2, j = 1, k = 3, u = 2$

$$S = \begin{bmatrix} 2 & 3 & 0 & 3 \\ 1 & 0 & 2 & 1 \\ 0 & 2^* & 1 & 1^* \\ 3 & 0 & 3 & 2 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 1 & 0^- & 1 & 2^+ \\ 1 & 1^+ & 1 & 1^- \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Third repetition: $i = 0, j = 3, k = 1, u = 3$

$$S = \begin{bmatrix} 2 & 3 & 0 & 3 \\ 1 & 0 & 2 & 1 \\ 0 & 2 & 1 & 1 \\ 3 & 2^* & 3 & 0^* \end{bmatrix} \quad C = \begin{bmatrix} 1 & 1^- & 1 & 1^+ \\ 1 & 0 & 1 & 2 \\ 1 & 2^+ & 1 & 0^- \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Forth repetition: $i = 1, j = 1, k = 3, u = 1$

$$S = \begin{bmatrix} 2 & 3 & 0 & 3 \\ 1 & 1^* & 2 & 0^* \\ 0 & 2 & 1 & 1 \\ 3 & 2 & 3 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0^- & 1 & 2^+ \\ 1 & 1^+ & 1 & 1^- \\ 1 & 2 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Fifth repetition: $i = 2, j = 3, k = 1, u = 2$

$$S = \begin{bmatrix} 2 & 3 & 0 & 3 \\ 1 & 1 & 2 & 0 \\ 0 & 1^* & 1 & 2^* \\ 3 & 2 & 3 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 1 & 2 \\ 1 & 2^+ & 1 & 0^- \\ 1 & 1^- & 1 & 1^+ \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Sixth repetition: $i = 0, j = 1, k = 3, u = 3$

$$S = \begin{bmatrix} 2 & 3 & 0 & 3 \\ 1 & 1 & 2 & 0 \\ 0 & 1 & 1 & 2 \\ 3 & 0^* & 3 & 2^* \end{bmatrix} \quad C = \begin{bmatrix} 1 & 1^+ & 1 & 1^- \\ 1 & 2 & 1 & 0 \\ 1 & 0^- & 1 & 2^+ \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

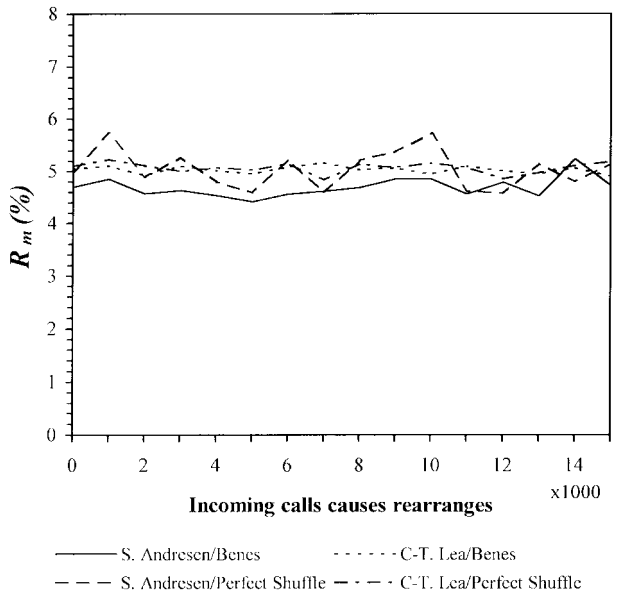


Fig. 6. The Andresen and Lea algorithm for the case of perfect shuffle and Benes network.

Seventh repetition: $i = 1, j = 3, k = 1, u = 1$

$$S = \begin{bmatrix} 2 & 3 & 0 & 3 \\ 1 & 0^* & 2 & 1^* \\ 0 & 1 & 1 & 2 \\ 3 & 0 & 3 & 2 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 2^+ & 1 & 0^- \\ 1 & 1^- & 1 & 1^+ \\ 1 & 0 & 1 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

If we continue the repetitions, the algorithm enters into a nonstop loop from the fifth to seventh repetition by swapping the same elements, $S(1,1) - S(1,3)$, $S(2,1) - S(2,3)$, $S(3,1) - S(3,3)$ and finally holes (zeroes) are created in the C matrix.

In order to avoid similar situations, we propose a simple correction technique: At any swapping of a pair we look back in the C matrix in order to ensure that no holes have been created. If a hole is found we choose this element and we continue the process leaving only satisfied columns to the left. The flow chart in Fig. 7 explains the Chiu–Siu modified algorithm in details. Our modification is indicated by the dashed box.

The application of this modification the previous example gives

$$P = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 11 & 14 & 0 & 12 & 8 & 2 & 6 & 4 & 3 \\ 9 & 10 & 11 & 12 & 13 & 14 & 15 & & \\ 7 & 5 & 9 & 13 & 1 & 15 & 10 & & \end{bmatrix}.$$

The initial S and C matrixes are

$$S = \begin{bmatrix} 2 & 3 & 0 & 3 \\ 2 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 3 & 0 & 3 & 2 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 2 & 0 & 0 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

First repetition: $i = 1, j = 0, k = 2, u = 1$

$$S = \begin{bmatrix} 2 & 3 & 0 & 3 \\ 1^* & 0 & 2^* & 1 \\ 0 & 1 & 1 & 2 \\ 3 & 0 & 3 & 2 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 1^+ & 1 & 1^- & 1 \\ 1^- & 0 & 1^+ & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

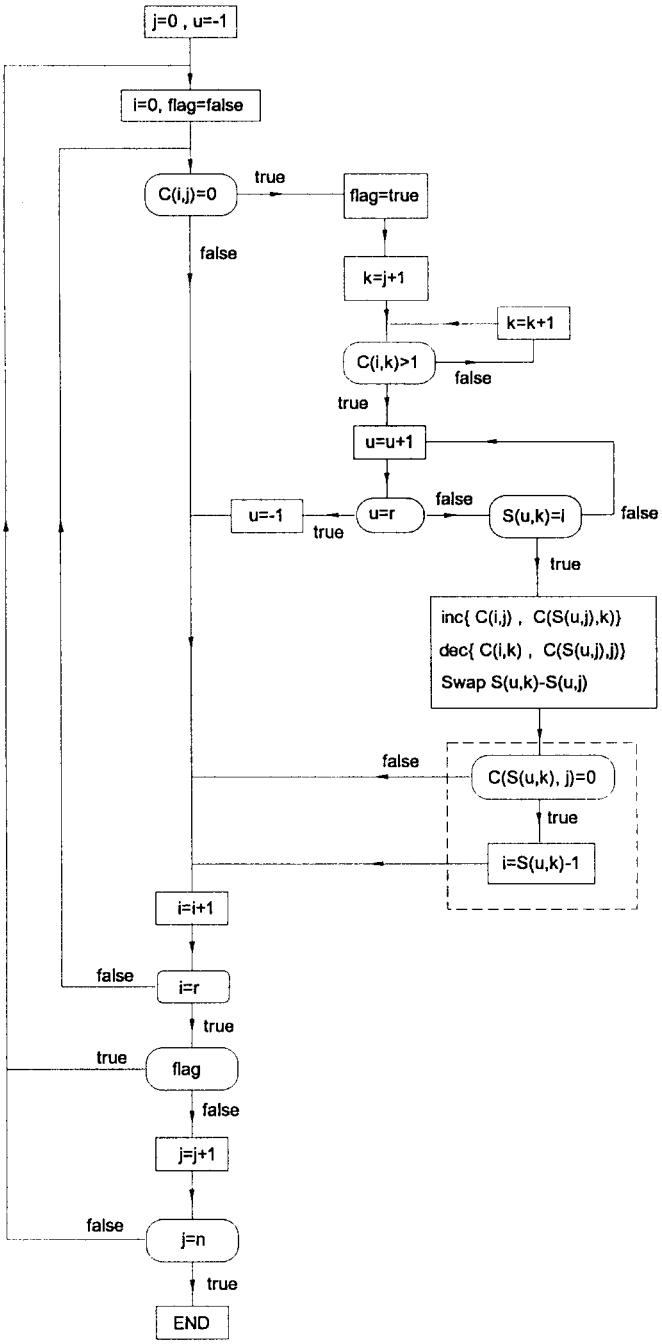


Fig. 7. Flow Chart of the Chiu–Siu modified algorithm.

Second repetition: $i = 2, j = 1, k = 3, u = 2$

$$S = \begin{bmatrix} 2 & 3 & 0 & 3 \\ 1 & 0 & 2 & 1 \\ 0 & 2^* & 1 & 1^* \\ 3 & 0 & 3 & 2 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 1 & 0^- & 1 & 2^+ \\ 1 & 1^+ & 1 & 1^- \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Third repetition: $i = 1, j = 1, k = 3, u = 1$

$$S = \begin{bmatrix} 2 & 3 & 0 & 3 \\ 1 & 1^* & 2 & 0^* \\ 0 & 2 & 1 & 1 \\ 3 & 0 & 3 & 2 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 1^- & 1 & 1^+ \\ 1 & 1^+ & 1 & 1^- \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

We observe in the third repetition that the algorithm jumps back in order to correct a hole that has been created in the

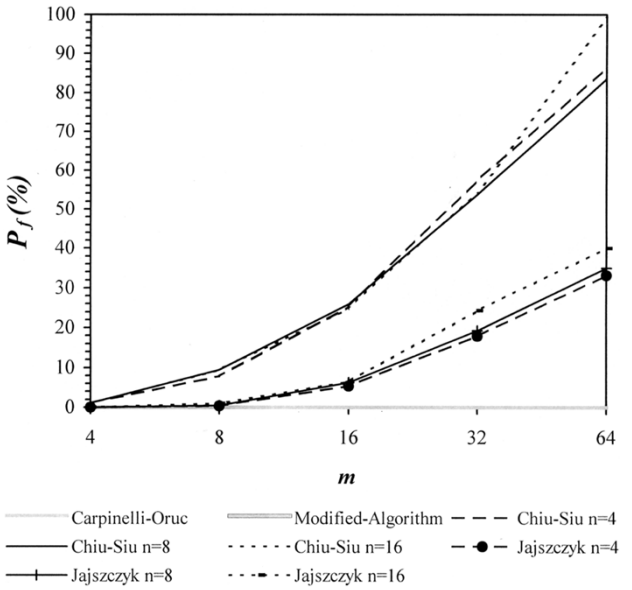


Fig. 8. Failure probability of the four control algorithms, as a function of the dimensions of the Clos Network (m, n).

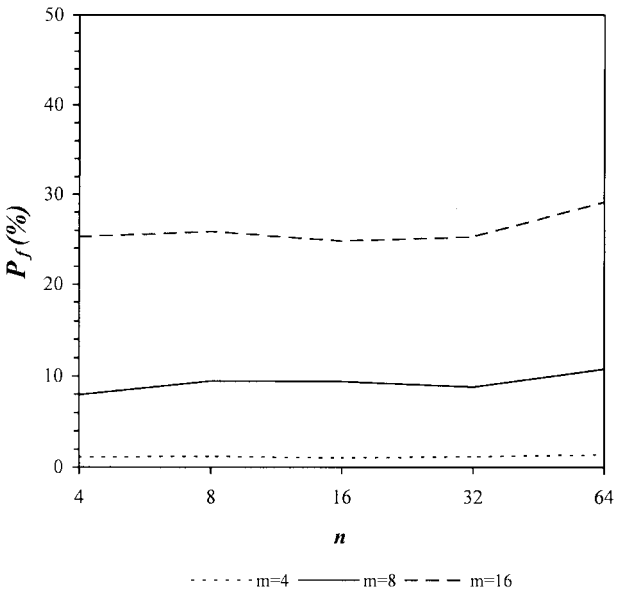


Fig. 9. Failure probability of the Chiu–Siu control algorithm according to the value of n.

position $C(2,2)$ after the second repetition. We have tested our algorithm in a series of S matrixes which failed to converge by algorithm [8], and we always obtained correct results.

APPENDIX B

We present an extended comparison of these algorithms using a well-known criterion, the failure probability P_f.

In Fig. 8 the failure probability of the four algorithms for various sizes of Clos networks is given. We observe an inverse exponential relationship of the failure probability with the network size. The Chiu–Siu algorithm has the maximum failure probability for large Clos Networks. The Carpinelli–Oru and our modification of the Chiu–Siu algorithm have zero failure probability as it was expected.

In the same figure the failure probability of the Chiu–Siu algorithm is studied in more detail. For every distinct value of n the factor m takes the values of X-axis. We observe that the failure probability is greatly affected by the value of m, in other words an increment to the number of the first-stage switches leads to a great performance degradation of the control algorithm. On the other hand, the behavior of the algorithm appears more stable to the variation of the number of inputs per first-stage switch (n value) as we can see in Fig. 9.

Jajszczyk algorithm shows a similar behavior for different values of n. The basic idea of this algorithm is the decomposition of the $m \times m$ matrix H. The sum of any row or column of H matrix equals n. As the number of zeros in the H matrix increases, the probability of a deadlock increases too. The number of zeros in the H matrix depends basically on the dimensions of H matrix and secondary to the value of n. In Fig. 8 the failure probability is given according to the value of n. We observe that as the n increases the failure probability increases slightly (more than Chiu–Siu algorithm). In other words, n does not effect very much the failure probability.

The next investigation concerns the application of backtracking method in the Jajszczyk algorithm. The results are

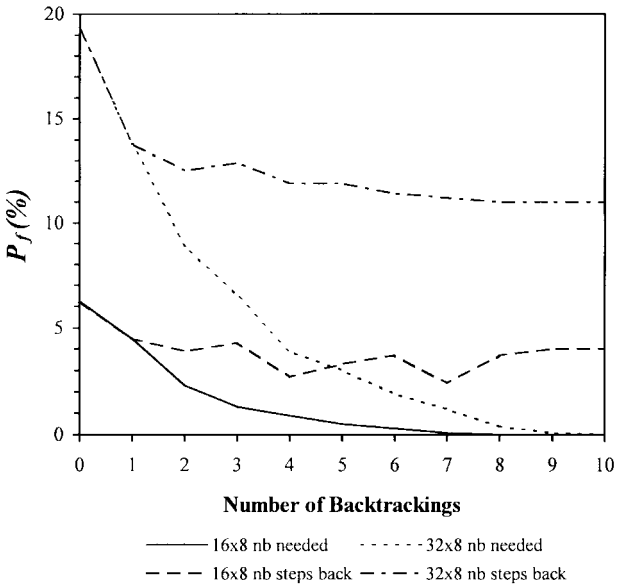


Fig. 10. Failure Probability of Jajszczyk algorithm with $n = 8$ and various values of m, for the two methods of backtracking.

shown in Fig. 10. We observe that as the size of the network increases the number of backtrackings required to decrease the failure probability increases too. However, the first method of backtracking, in which we go back as many steps as needed to find a solution, never fails and the algorithm manages to eliminate the failure probability even if the network size increases. If we apply the second method of backtracking, going back fixed number of steps, the results are different as we observe in Fig. 10. The failure probability can not reach the zero value, because the algorithm slides from the region where a wrong element had been selected and remains to a significant level. If the size of the network is increased the performance of the algorithm becomes worse as it can be seen from the same figure.

For Benes-type networks no failure probability is applicable since they work on self routing logic.

REFERENCES

[1] S. Andresen, "The looping algorithm extended to base 2t rearrangeable switching networks," *IEEE Trans. Commun.*, vol. COM-25, no. 10, pp. 1057–1063, Oct. 1977.

[2] Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic*. New York: Academic, 1965.

[3] J. D. Carpinelli and A. Y. Oru, "A nonbacktracking matrix decomposition algorithm for routing on clos networks," *IEEE Trans. Commun.*, vol. 41, no. 8, pp. 1241–1251, Aug. 1993.

[4] Y. K. Chiu and W. C. Siu, "Comment: Novel algorithm for clos-type networks," *Electron. Lett.*, vol. 27, no. 6, pp. 524–526, Mar. 1991.

[5] J. Gordon and S. Srikanthan, "Novel algorithm for clos-type networks," *Electron. Lett.*, vol. 26, no. 21, pp. 1772–1774, Oct. 1990.

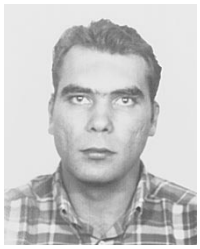
[6] S.-T. Huang and S. K. Tripathi, "Finite state model and compatibility theory: New analysis tools for permutation networks," *IEEE Trans. Comput.*, vol. C-35, no. 7, pp. 591–601, July 1986.

[7] J. Y. Hui, *Switching and Traffic Theory for Integrated Broadband Networks*. Norwell, MA: Kluwer, 1990.

[8] A. Jajszczyk, "A simple algorithm for the control of rearrangeable switching networks," *IEEE Trans. Commun.*, vol. COM-33, no. 2, pp. 169–171, Feb. 1985.

[9] C.-T. Lea, "Bipartie graph design principle for photonic switching systems," *IEEE Trans. Commun.*, vol. 38, no. 4, pp. 529–538, Apr. 1990.

[10] C.-S. Raghavendra and A. Varma, "Rearrangeability of the five-stage shuffle/exchange network for $N = 8$," *IEEE Trans. Commun.*, vol. COM-35, no. 8, pp. 808–812, Aug. 1987.



Ioannis Gragopoulos (S'95) was born in Thessaloniki, Greece, in 1969. He received the diploma in electrical engineering from the Aristotle University of Thessaloniki, Greece, in 1993. His diploma thesis was on routing and flow control algorithms in integrated networks. Now he is pursuing the Ph.D. degree in ISDN networks at the Aristotle University of Thessaloniki, Greece.

His research interests include ATM switching techniques, digital telephony, routing algorithms, and queueing theory.

Fotini-Niovi Pavlidou (M'87) received the diploma in electrical engineering and the Ph.D. degree in computer networks from the Aristotle University of Thessaloniki, Greece, in 1979 and 1987, respectively.

From 1980 to the present, she has taught several courses on digital telecommunications, mobile telephony, and computer networks. She is presently an Assistant Professor in the Section of Telecommunications. Her research interests include queueing theory, ATM switching techniques, routing algorithms, and multiple access in wireless communications.