# Computer Programs and Examples in the study of Central Groupoids, Central Digraphs, and Zero-One Matrices $A$ Satisfying $A^2 = J$

Frank Curtis, John Drew, Chi-Kwong Li
Department of Mathematics, College of William and Mary,
PO Box 8795, Williamsburg, VA 23187-8795
E-mail: fecurt@math.wm.edu, jhdrew@math.wm.edu, ckli@math.wm.edu
and
Dan Pragel
Department of Mathematics, SUNY at Albany,
Albany, NY 12222
E-mail: rundmp@rochester.rr.com

September 5, 2003

We have established five Matlab programs to facilitate the study of central digraphs. They are stored at:

`http://www.resnet.wm.edu/~cklixx/matrixlib.html`

The following programs can be found there:

1. MakeStandard($k$)

   This program creates the standard matrix in $\mathcal{A}_{k^2}$.

2. MakeSwitch($p, q, r, s, A$)

   This program performs the switch $[p, q; r, s]$ on a given $A \in \mathcal{A}_n$ as described in Section 3 if the switch is legal.

3. AllRanks($k$)

   This program creates matrices in $\mathcal{A}_{k^2}$ of all

   possible ranks.

4. DRM($A$)

   This program operates on a matrix $A \in \mathcal{A}_n$ and produces a permutationally similar matrix that exhibits "decreasing-row-multiplicity", i.e., rows with higher multiplicity

are above rows with lower multiplicity and identical rows are adjacent. We present the following example of the transformation of the matrix $A \in A_9$ into $DRM(A)$:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \; DRM(A) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

5. PerSim$(A, B)$

   This program determines whether two matrices $A, B \in \mathcal{A}_n$ are permutationally similar.

   In general, it is difficult to determine whether two zero-one matrices are permutationally similar. However, due to the special properties associated with the matrices in $\mathcal{A}_n$ (e.g. constant row and column sums, sets of equal rows and columns, etc.), we are able to drastically reduce the number of necessary comparisons between matrices $A$ and $B$ that may or may not be similar. Preliminary tests such as verifying that the matrices have the same rank and have equal row and column multiplicity vectors are not sufficient to determine permutation similarity, but can be used initially in the program logic to show that certain matrices are not permutationally similar. If these tests are inconclusive, the problem then reduces to performing all possible permutations on the first matrix, a process that can be facilitated with special consideration of the row and column multiplicity vectors.

   The computer program PerSim with the above considerations was implemented and utilized throughout our study of matrices in $\mathcal{A}_n$. Given two matrices $A$ and $B$, preliminary testing was used to determine that the matrices have the same size, rank, multiplicity vectors, and that both in fact correspond to central groupoids. If the matrices were still possibly similar, then both were permuted into decreasing-row-multiplicity form and all possible permutations of the matrix $A$ were considered. Under normal circumstances, a total of $n!$ permutations are necessary to exhaust all permutations of an $n \times n$ matrix. However, in decreasing-row-multiplicity form, one needs only to consider permutations within each set of equal rows and between sets of equal size. For example, given the matrix $A$ above, we have a row multiplicity vector of $\{3, 2, 2, 1, 1\}$ and a column multiplicity vector of $\{3, 2, 2, 1, 1\}$. If a matrix $B$ has the same multiplicity vectors, then one must only consider permutations within the three equal rows of $A$, within each of the two pairs of equal rows in $A$, and all further permutations after each possible swap of sets of rows of equal size. Consequently, the total number of permutations that need to be considered is $(3!)(2!)(2!)(2!)(2!) = 96$. When compared to $n! = 9! = 362880$ possible permutations, the program run-time has been considerably reduced.