ITERATIVE DECOMPOSITION OF PERMUTATIONS

Z. Kokosiński

Instytut Elektrotechniki i Elektroniki Politechnika Krakowska,
Kraków, Poland

Received July 22, 1988

Abstract

Given symmetric group S_n and its subgroup $(S_q \times S_p)$, $p + q = n$, we establish necessary and sufficient conditions for the decomposition of S_n into left cosets of $(S_q \times S_p)$ in S_n. If $n = 2^m$, $(m = 1,2, \ldots)$ and $p = q = n/2$ then by iteration we obtain a decomposition of an arbitrary $\pi \in S_n$ in the form $\pi = (\beta_1^0)(\beta_2^1, \beta_2^0) \ldots (\beta_N^{n/2-1}, \ldots, \beta_N^0)$, where $N = \log_2 n$ and βi is the j-th left coset leader obtained on the i-th stage. We develop an $0(n \log n)$ serial algorithm for the programming the tree cellular permutations networks which result from the above decomposition scheme.

1. Introduction

An important problem in the theory of interconnection networks is to find methods of decomposing an arbitrary permutation of a large number of elements. Basing on those methods various permutation networks as well as control algorithms are to be designed.

Opferman and Tsao-Wu published an algorithm for decomposing an arbitrary permutation of $n = d \times q$ elements into d permutations of q elements each and $(2q - 1)$ permutations of d elements each [7]. Their algorithm is devoted to a classical Benes network [1], which is a member of Clos' type networks [3]. A modified version of that algorithm was presented by Ramanujam [10]. However, Kubale showed it to be only partially correct. He pointed out that for this class of interconnection networks the most efficient approach is a decomposition by edge coloring bipartite multigraphs.

Kautz et al. [5] described another large class of interconnection networks: cellular permutation networks. This class consists of several families of networks built from 2-permuters (i.e. triangular, diamond, rectangular, pruned rectangular, rhomboidal, almost square etc.). Oruc and Prakash [8] developed control algorithms for triangular and diamond cellular arrays based on a certain factorization of permutation cycles. Recently Oruc and Oruc found the best known algorithm for the programming triangular permutation arrays through iterative coset decomposition of symmetric groups [9].

The method of iterative decomposition of permutations presented below is to be applied for the programming a family of cellular permutation networks, we call tree permutation arrays. It is based on the decomposition of symmetric groups into cosets. An arbitrary permutation $\pi \in S_n$ ($n = 2^m$; $m = 1, 2, \ldots$) is transformed by the algorithm into a composition of a certain n-permutation $\beta \in S_n$ being a left coset leader of the subgroup $(S_p \times S_q)$ in S_n ($p = q = n/2$) and a pair of two $n/2$-permutations $(\pi_1, \pi_2) \in (S_p \times S_q)$. This decomposition procedure requires $0(n)$ sequential time and is repeated iteratively stage by stage $N = \log_2 n$ times. Hence, the time complexities are $0(n \log n)$ for the serial algorithm and $0(n)$ for the parallel algorithm, respectively.

2. Validity of decomposition

From the theory of groups the following elementary results form a background for developing the decomposition procedure.

Theorem 1. Let G be a permutation group on a finite set $\Omega = \{1, 2, \ldots, n\}$. Let P be any proper subset of Ω. Then the permutations of G fixing all the elements of P form a subgroup K. The permutations permuting the elements of P among themselves form a group H which contains K as a normal subgroup.

Theorem 2. A group G is isomorphic to the direct product of two subgroups A and B (denoted $A \times B$) if A and B are normal subgroups such that $A \cap B = 1$, $A \cup B = G$.

Lemma 1. Let $G = S_n$ and $\Omega = P \cup Q$, $P \cap Q = \emptyset$. Let $|\Omega| = n$, $|P| = p$, $|\Omega| = q$ and $p, q \neq 0$. If according to Theorem 1 the permutations of S_n fixing all elements of $P(Q)$ form a normal subgroup K_1 (K_2), $K_1 \leq H_1 \leq S_n$ $(K_2 \leq H_2 \leq S_n)$, then $H_1 = H_2 = K_1 \times K_2$.

Proof. Proof follows from Theorems 1 and 2.

It is clear that $K_1(K_2)$ is isomorphic to the symmetric group $S_q(S_p)$.

For given group G and its subgroup H the set of all elements $\{gh;\ g \in G,$ g fixed, $h \in H\}$ is called a left coset of H in G.

Lemma 2. Two different left cosets of H in G are either disjoint or identical sets of elements.

We write $G = g_1 H \cup g_2 H \cup \ldots \cup g_l H$, where l denotes the number of left cosets of H in G and g_i $(i = 1, 2, \ldots, l)$ we call the left coset leader.

Let g_1 and g_2 be any two permutations in S_n. They may always be written in the following two-row matrices:

$$g_1 = \begin{vmatrix} \{\ P\ \}, \{\ Q\ \} \\ \{g_1(P)\}, \{g_1(Q)\} \end{vmatrix} = \left| \underbrace{\begin{matrix} \{\ P - P_1\ \}, \{\ Q_1\ \}, \\ \{g_1(P - P_1)\}, \{g_1(Q_1)\}, \end{matrix}}_{\{P\}} \underbrace{\begin{matrix} \{\ Q - Q_1\ \}, \{\ P_1\ \} \\ \{g_1(Q - Q_1)\}, \{g_1(P_1)\} \end{matrix}}_{\{Q\}} \right| =$$

$$= \begin{vmatrix} \{g_1^{-1}(P)\}, \{g_1^{-1}(Q)\} \\ \{\ P\ \}, \{\ Q\ \} \end{vmatrix}$$

$$g_2 = \left|\begin{matrix}\{\ P\ \}, \{\ Q\ \} \\ \{g_2(P)\}, \{\ g_2(Q)\}\end{matrix}\right| = \left|\underbrace{\begin{matrix}\{\ P - P_2\ \}, \{\ Q_2\ \}, \\ \{g_2(P - P_2)\}, \{g_2(Q_2)\},\end{matrix}}_{\{P\}} \underbrace{\begin{matrix}\{\ Q - Q_2\ \}, \{\ P_2\ \} \\ \{g_2(Q - Q_2)\}, \{g_2(P_2)\}\end{matrix}}_{\{Q\}}\right| =$$

$$= \left|\begin{matrix}\{g_2^{-1}(P)\}, \{g_2^{-1}(Q)\} \\ \{\ P\ \}, \{\ Q\ \}\end{matrix}\right|.$$

The elements of $P - P_1$ and $Q_1(P - P_2$ and $Q_2)$ are moved under $g_1(g_2)$ into elements of P. The elements of $Q - Q_1$ and $P_1(Q - Q_2$ and $P_2)$ are moved under $g_1(g_2)$ into elements of Q.

Theorem 3. Two left cosets of H in G: g_1H and g_2H are disjoint if the following condition holds

$$\{(P_1 \neq P_2) \cup (Q_1 \neq Q_2)\}.$$

Proof. Proof is provided by contradiction. The above condition reflects an obvious fact that no two cosets can be identical if the permutations which belong to those cosets, map different elements from P into Q or from Q into P. The dependent condition $\{[(P - P_1) \neq (P - P_2)] \cup [(Q - Q_1) \neq (Q - Q_2)]\}$ reflects an analogous observation related to complementary subsets $(P - P_1)$ and $(Q - Q_1)$.

The above mathematical background allows us to develop a recursive decomposition procedure in the next section.

3. Algorithm

Let $n = 2^m$ $(m = 1, 2, \ldots)$ and $p = q = n/2$. As we have stated previously an arbitrary permutation $\pi \in S_n$ is factorized by the algorithm iteratively, stage by stage.

On the i-th stage $(i = 1, 2, \ldots, N;\ N = \log_2 n)$ $2^{(i-1)}$ permutations $\pi_i^j(j = 0, \ldots, 2^{(i-1)} - 1)$ is decomposed by a recursive procedure into 2^i permutations forming $(i + 1)$-th stage. This iteration may be written as follows

$$\pi_i^j = \beta_i^j(\pi_{i+1}^{2j+1}, \pi_{i+1}^{2j}).$$

The lower index i denotes the number of a stage, the upper index j denotes the number of the permutation on the i-th stage (see Fig. 1).
The set of all left coset leaders β_i^j obtained on subsequent stages of the decomposition produces a full binary tree shown in Fig. 2. The upper index j has a decimal value corresponding to the binary path in the tree.

The final decomposition of the permutation π_1^0 is described by the equation

$$\pi_1^0 = (\beta_1^0)(\beta_2^1, \beta_2^0) \ldots (\beta_N^{n/2-1} \ldots \beta_N^0).$$

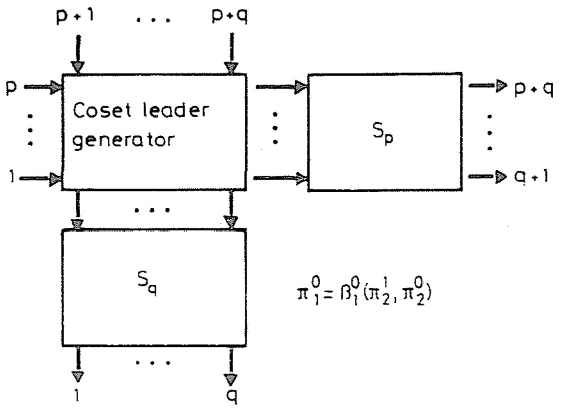

Fig. 1. Decomposition scheme for $i = 1, j = 0$

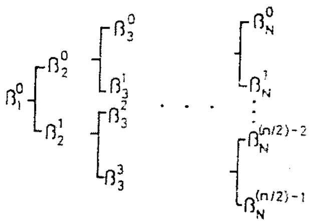

Fig. 2. A full binary tree of left coset leaders β_i^j

Algorithm 1

Input: n, π_1^0 in two-row matrix form; the first row is called PERM1, the second row is called PERM2.

Output: table I, in which [T(1, k), T(2, k)] denotes k-th transposition (cell in the "bend" state).

```
1. $i \leftarrow 1$;
2. $j \leftarrow 0$;
3. $t \leftarrow 0$;
4. call algorithm TREEFACTOR ($n, i, j$, PERM1, PERM2, $t$, $T$).
```

algorithm TREEFACTOR (n, i, j, PERM1, PERM2, t, T).

```
1. $a \leftarrow n/2^i$;
2. relabel PERM1, PERM2 according to the following pattern:
   2.1. PERM1 $\leftarrow |1, \ldots, a, a+1, \ldots, 2a|$;
   2.2. PERM2 $\leftarrow |\pi_i^j(1), \ldots, \pi_i^j(a), \pi_i^j(a+1), \ldots, \pi_i^j(2a)|$;
3. $c \leftarrow aj$;
   compute correction
4. $u \leftarrow 0$;
```

```
5. $v \leftarrow 0$;
6. for $k \leftarrow 1$ until a do
   6.1. if PERM2 $(k) \leq a$ then
        6.1.1. $u \leftarrow u + 1$;
        6.1.2. $T(1, t + u) \leftarrow$ PERM2 $(k) + c$;
   6.2. if PERM2 $(a + k) > a$ then
        6.2.1. $v \leftarrow v + 1$;
        6.2.2. $T(2, t + v) \leftarrow$ PERM2 $(a + k) + c$;
7. if $(i + 1) > \log_2 n$ then return
   else
   7.1. PERM1 $\leftarrow |a + 1, a + 2, \ldots, 2a, 1, 2, \ldots, a|$;
   7.2. for $1 \leftarrow 1$ until $v$ do
        7.2.1. PERM1 $(T(2, t + 1) - c - a) \leftarrow T(1, t + 1)$,
        7.2.2. PERM1 $(T(1, t + 1) - c + a) \leftarrow T(2, t + 1)$;
   7.3. $t \leftarrow t + v$;
   7.4. for $k \leftarrow 1$ until a do
        7.4.1. PERM1A $(k) \leftarrow$ PERM1 $(a + k)$;
        7.4.2. PERM2A $(k) \leftarrow$ PERM2 $(a + k)$;
        7.4.3. PERM1B $(k) \leftarrow$ PERM1 $(k)$;
        7.4.4. PERM2B $(k) \leftarrow$ PERM2 $(k)$;
   7.5. call algorithm TREEFACTOR $(n, i + 1, 2j$, PERM1A, PERM2B, $t, T)$;
   7.6. call algorithm TREEFACTOR $(n, i + 1, 2j + 1$, PERM1B, PERM2B
        $t, T)$;
8. return.
```

The time complexity of above sequential algorithm on the i-th stage is $0(n)$: $2^{(i-1)}$ left coset leaders of length $n/2^{(i-1)}$ are generated. Since $N = \log_2 n$ iterations are performed, the total time complexity is $0(n \log n)$. The parallel algorithm consumes $0(n)$ time.

4. Tree permutation arrays

In Fig. 3 two realizations of tree permutation arrays are visualised. Those cellular permutation networks have different assignments of inputs and outputs. For given mapping of inputs into outputs the first network is programmed with $\pi_1^0 = \pi$, while the second one is programmed with $\pi_1^0 = [\pi_{\text{rev}}]^{-1}$. This last solution is preferable from practical point of view, because it allows overlapping of the set up time of the network and the propagation delay. Moreover, there is no conflict between the marking of cell transpositions and the whole network input-output marking in the case of the second network. A primary network setting is that of "cross" state. When Algorithm 1 is applied, all cells specified in the table T are programmed to the "bend" state

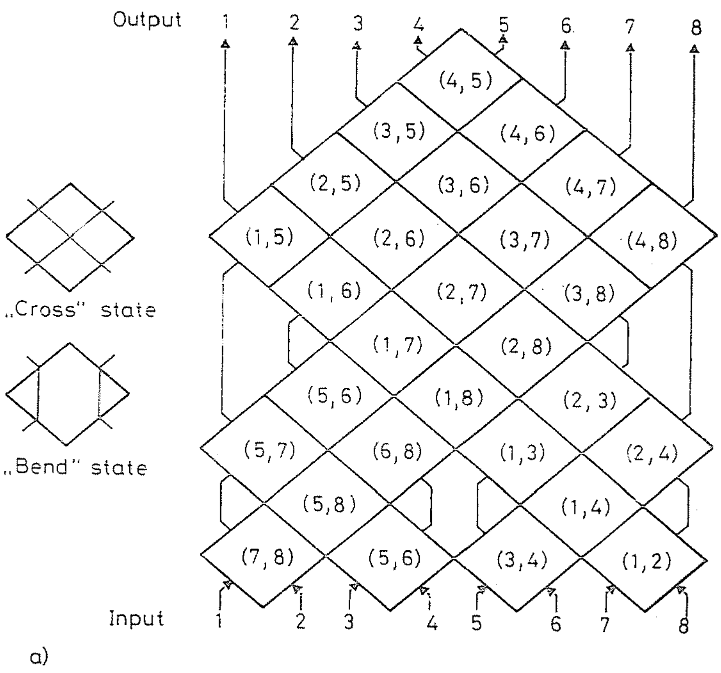

Fig. 3a. Tree permutation array for $\pi_1^0 = \pi$; $n = 8$

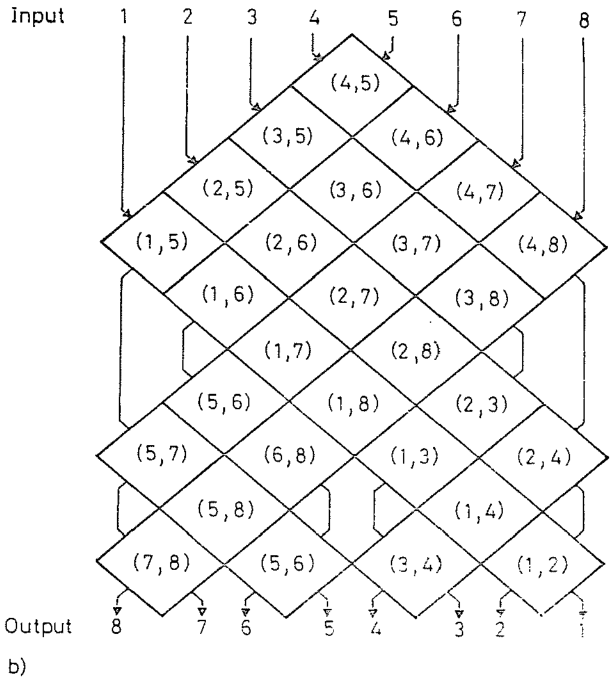

Fig. 3b. Tree permutation array for $\pi_1^0 = [\pi_{\text{rev}}]^{-1}$; $n = 8$

References

1. Beneš, V. E.: Permutation groups, complexes, and rearrangeable connecting networks BSTJ, *43*, No. 4 (1964), pp. 1619–1640.
2. Beneš, V. E.: Mathematical theory of connecting networks and telephone traffic. Academic Press, New York 1965.
3. Clos, C.: A study of non-blocking switching networks. BSTJ, *32*, No. 2 (1953), pp. 406–424.
4. Hall, M.: The theory of groups. MacMillan, New York 1959.
5. Kautz et. al.: Cellular interconnection arrays. IEEE Trans. Comp., *C—17*, No. 5 (1968), pp. 443–451.
6. Kubale, M.: Comments on "Decomposition of permutation networks" IEEE Trans. Comp., *C—31*, No. 3 (1982), pp. 265.
7. Opferman, D. C., Tsao-Wu, N. T.: On a class of rearrangeable switching networks. BSTJ, *50*, No. 3. (1971), pp. 1579–1618.
8. Oruc, Y. A., Prakash, D.: Routing algorithms for cellular interconnection arrays. IEEE Trans. Comp., *C—33*, No. 8 (1984), pp. 939–942.
9. Oruc, Y. A., Oruc, Y. M.: Programming cellular permutation networks through decomposition of symmetric groups. IEEE Trans. Comp., *C—36*, No. 7 (1987), pp. 802–809.
10. Ramanujam, H. R.: Decomposition of permutation networks. IEEE Trans. Comp., *C—22*, No. 7 (1973), pp. 639–643.

Zbigniew Kokosiński Kraków, Poland