# PERMUTATION ROUTING VIA CAYLEY GRAPHS WITH AN EXAMPLE FOR BUS INTERCONNECTION NETWORKS

GENE COOPERMAN AND LARRY FINKELSTEIN

Abstract. Cayley graphs have been used extensively to design interconnection networks and provide a natural setting for studying point-to-point routing [1, 2, 3, 5, 6, 7, 12]. The extension of these techniques to the more important problem of permutation routing on interconnection networks presents fundamental problems. This is due to the potentially explosive growth in both the size of the graph and the number of generating permutations, referred to as *one-step permutation routes*, used to define the underlying graph. This paper describes a technique for moderating that growth so that the techniques in [8] can be applied for finding optimal permutation routes. In a particularly striking example, a bus interconnection architecture involving $1.0 \times 10^{17}$ permutations (nodes of the Cayley graph) is reduced to a computation on a graph with only 3,950 nodes. Further, it is shown how many of the 58,624 generators (directed edges labelled by one-step permutation routes) at each node of the graph may be eliminated as locally redundant.

## 1. Introduction

There has been an extensive literature on the use of Cayley graphs to design interconnection networks [1, 2, 5, 6, 7, 12]. Cayley graphs have been used extensively to study point-to-point routing, and they have been particularly attractive for the degree-diameter problem [3, 5, 7]. Nevertheless, other patterns of routing, such as broadcast, permutation routing, and general many-to-one routing are at least as important for parallel computing. This paper describes the use of Cayley graphs to study permutation routing. There are also other approaches to permutation routing [1, 11] that concentrate more on heuristics and an overall framework.

A *Cayley graph* $\mathcal{G}$ is a directed labelled graph associated with a group, $G$, and generating set $S$. The nodes of $\mathcal{G}$ are elements of $G$ and the edges are labelled by generators in $S$. Two nodes, $g$ and $h$ are connected by a directed edge, $(g, h)$, and the edge is labelled by $s \in S$, if $h = gs$. In the case where $S = S^{-1}$, whenever $g$ is connected to $h$ by a directed edge labelled by $s$, $h$ is simultaneously connected to $g$ by a directed edge labelled by $s^{-1}$, and so $\mathcal{G}$ may be viewed as undirected. This is typically the case and will be assumed throughout this paper. Cayley graphs

1991 *Mathematics Subject Classification*. Primary 05C25, 68M10; Secondary 20-04, 68R10, 90B12.

*Key words and phrases.* permutation routing, Cayley graph, bus interconnection network.

The authors were partially supported by NSF Grant CCR-9204469.

have the property that they are *vertex-symmetric*. This means that for any two nodes, there is a graph isomorphism of the graph into itself that maps the first node into the second. In the case of Cayley graphs, this graph isomorphism also preserves the labels of the edges. In fact, this property can be easily established by simply embedding $G$ into $\mathrm{Aut}(\mathcal{G})$ through the left regular action so that the resulting image is transitive on the nodes of the graph and preserves edge labels.

A spanning tree $\mathcal{T}$ for $\mathcal{G}$ rooted at the identity $e$ allows one to find shortest paths between arbitrary nodes. Given elements $g$ and $h$ of $G$, one uses $\mathcal{T}$ to find a shortest path from $e$ to $g^{-1}h$. If $w$ is the word in the edge labels along this path, then $h = gw$. Since $\mathcal{G}$ is vertex-symmetric, it follows that $w$ also defines a shortest path in $\mathcal{G}$ from $g$ to $h$ as well. An important interpretation is that $w$ is a minimal length word in the generating set $S$ that represents the group element $g^{-1}h$. In particular, this allows for the solution of the *minimal word* problem for $G$ relative to the generating set $S$.

In [8], an efficient method of encoding Cayley graphs was developed, using standard techniques from computational group theory [13]. This method required $\log_2 3$ bits of storage per node of the graph to store a data structure for a spanning tree for the Cayley graph. The spanning tree was derived essentially through a method of breadth-first search. During computation of the data structure, $\log_2 5$ bits per node were temporarily required. Since this is approximately 2 bits, it was called the 2-bit method. Furthermore, it was shown how to use this data structure to find shortest paths in $\mathcal{G}$ between any two nodes. Applications of this method are presented in [9].

We will now show how to apply these ideas to permutation routing in an arbitrary interconnection graph $\mathcal{I}$. We first define a *one-step permutation route* $\pi$ to be a permutation of the nodes $\Omega$ of $\mathcal{I}$ with the property that either $\pi(x)$ is connected to $x$ for each node $x$ or $\pi(x) = x$. The idea is that any one-step permutation route can be executed in parallel without any conflicts. A $k$-step permutation route is one that is obtained from a sequence of at most $k$ one-step permutation routes. If $\mathcal{I}$ is connected, then a transposition of the nodes of any edge is a one-step permutation route. This leads to the following observation.

**Theorem 1.** *If $\mathcal{I}$ is connected, then the set $S$ of one-step permutation routes generates $\mathrm{Sym}(\Omega)$ and any permutation of the nodes of $\Omega$ is a $\binom{|\Omega|}{2}$ permutation route.*

The basic problem we want to consider is given an arbitrary permutation $\pi$ of $\mathcal{I}$, express $\pi$ as a sequence of $k$ one-step permutation routes where $k$ is minimal. This can then be viewed as finding a word of minimal length in the set $S$ of one-step permutations that represents $\pi$. It is precisely this problem which must be solved.

## 2. Bus Interconnection Networks

We now apply the previous ideas to bus interconnection networks. From this, it will be clear how to generalize to other examples. Specifically, let $\mathcal{C}$ be a set of chips and let $\mathcal{B}$ be a set of bus lines. Let $\Omega = \mathcal{C} \cup \mathcal{B}$ and let $\mathcal{I}$ be a graph on $\Omega$ such that $(a, b)$ is an edge of $\mathcal{I}$ if and only if $a \in \mathcal{C}$, $b \in \mathcal{B}$, and chip $a$ is connected to bus line $b$.

The following model corresponds to the case in which each chip has only one register. In each clock cycle, for each chip $a$, either the data in $a$ is interchanged

with the data on some unique bus line $b$ or else the data on chip $a$ remains fixed. Section 5 discusses how to generalizes this to multiple registers per chip. Essentially, one identifies $\mathcal{C}$ with the set of chip registers, rather than chips. The simpler model is analyzed in detail in section 3 to illustrate the concepts.

Let $S$ be the set of all involutions $g \in \mathrm{Sym}(\Omega)$ with the property that $g$ is a product of disjoint transpositions of the form $(a\ b)$ where $a \in \mathcal{C}$ and $b \in \mathcal{B}$. The elements of $S$ are precisely the one-step permutation routes of $\mathcal{I}$. Let $M = \max(|\mathcal{C}|, |B|)$ and let $m = \min(|\mathcal{C}|, |\mathcal{B}|)$. Then, the size of $|S|$ is bounded above by $(M+1)^m$.

This is a restricted version of an interconnection network in which distinct data is both read and written to the bus in one clock cycle, thus insuring that the communication is indeed a permutation. This may be the physical situation if the setup time to connect chips and bus lines dominates the actual transfer time. The bus line and/or the chip can hold dummy data, in which case the important transfer of data is in one direction only.

## 3. Experimental Model

For simplicity in computational experiments, we also assume that each chip is connected to each bus line. This allows us to find experimental data dependent only on parameters $|\mathcal{C}|$ and $|\mathcal{B}|$. It is easy to see that if $S$ is the set of one-step permutation routes for $\mathcal{I}$ defined in the previous section, then $S$ generates $G = \mathrm{Sym}(\mathcal{C} \cup \mathcal{B})$. The technique can be easily applied to other interconnection patterns, and details of that generalization are contained in section 5.

The goal is to express each permutation route as a shortest possible sequence of one-step permutation routes. In the language of group theory, for each permutation $g$ of $\Omega$ that setwise stabilizes $\mathcal{B}$, we must find a shortest word in $S$ that represents $g$. Given the Cayley graph $\mathcal{G}$ for $G$ with generating set $S$, this is equivalent to finding a shortest path in $\mathcal{G}$ from the identity node 1 to the node $g$.

Rather than work with the Cayley graph $\mathcal{G}$, we will work with a reduced labelled multigraph, defined below. This smaller graph greatly reduces the necessary computation. The reduced graph is constructed through two reductions. First, we construct a Cayley coset graph $\mathcal{G}_H$ from $\mathcal{G}$, and second, we construct a reduced labelled multigraph $\tilde{\mathcal{G}}$ from $\mathcal{G}_H$.

The first reduction is obtained by noting that we are concerned only with permutations of data on the chips, and not with permutations of whatever data may initially be on the bus lines. This can be best expressed by right cosets $Hg \in G/H$, where $g \in G$ and $H \cong \mathrm{Sym}(\mathcal{B})$ is the subgroup of $G$ that fixes $\mathcal{C}$ pointwise. Formally, the *Cayley coset graph* of $G/H$ with generating set $S$ is a directed labelled multigraph $\mathcal{G}_H$, with nodes labelled by $G/H$. Two nodes $Hg_1$ and $Hg_2$ are connected by an edge with label $s$ if $Hg_1 s = Hg_2$. Thus, the image of $g' \in Hg$ on $\mathcal{C}$ depends only on $g$ and not on the choice of $g'$. So, it suffices to find a shortest word $w$ in $S$ representing an element of $Hg$. Many routing problems can be formulated in terms of Cayley coset graphs (also referred to as group action graph in [2]). In general, Cayley coset graphs are not vertex-symmetric.

The second reduction is obtained from the symmetries of the chosen bus interconnection architecture. Recall that a *group automorphism* is a permutation $\sigma$ of $G$ such that $\sigma(gh) = \sigma(g)\sigma(h)$ and $\sigma(g^{-1}) = (\sigma(g))^{-1}$ for all $g, h \in G$. For $G$

corresponding to the bus interconnection architecture $\mathcal{I}$, we define a *symmetry* to be an automorphism, $\sigma$, of $G$ such that $\sigma(H) = H$ and $\sigma(S) = S$.

This definition insures that the symmetries of $G$ act as graph automorphisms of the Cayley graph $\mathcal{G}$ (with generating set $S$), which map paths of minimal length in $\mathcal{G}$ to other paths of minimal length. If $(g_1, g_2)$ is a directed edge of $\mathcal{G}$ with label $s \in S$, then $\sigma$ maps $(g_1, g_2)$ to the directed edge $(\sigma(g_1), \sigma(g_2))$ with label $\sigma(s) \in S$. This and the invertibility of $\sigma$ imply that $\sigma$ preserves the paths of minimal length in $\mathcal{G}$. Furthermore, as indicated above, we are actually interested in shortest words which represent a fixed coset $Hg$. But the symmetries of $G$ also act as graph automorphisms of $\mathcal{G}_H$ in the same sense as for $\mathcal{G}$. To see this, note that if $(Hg_1, Hg_2)$ is an edge of $\mathcal{G}_H$ with edge label $s \in S$ then $\sigma(H) = H$ and $\sigma(s) \in S$ implies that $(\sigma(Hg_1), \sigma(Hg_2)) = (H\sigma(g_1), H\sigma(g_2))$ is an edge of $\mathcal{G}_H$ with label $\sigma(s)$. Thus $\sigma$ preserves paths of minimal length in $\mathcal{G}_H$ as well.

Conjugation provides a natural means of constructing such symmetries. Define the *conjugate* of $g$ by $u$ to be $g^u = u^{-1}gu$, and extend the definition to sets $H^U = \{h^u\colon\ h \in H, u \in U\}$. The mapping $\sigma_u\colon\ g \mapsto g^u$ is an automorphism of $G$, and is known in group theory as an *inner automorphism*. In the case of our bus interconnection network, we let $U \cong \mathrm{Sym}(\mathcal{C}) \times \mathrm{Sym}(\mathcal{B})$, and choose as the set of symmetries $\{\sigma_u\colon\ u \in U\}$. We often refer to a symmetry $\sigma_u$ simply as $u$. Note that $H^U = H$ and $S^U = S$ for this choice of $U$, and so the definition of symmetry is satisfied. Hence, conjugation by $U$ preserves shortest words. From group theory, it is known that if $|\mathcal{C} \cup \mathcal{B}| \neq 6$, then the set $U$ contains all possible symmetries.

With this motivation, we now introduce the *reduced labelled multigraph* $\tilde{\mathcal{G}}$, derived from $\mathcal{G}$, in which nodes of $\tilde{\mathcal{G}}$ are identified with subsets $[Hx] = (Hx)^U = Hx^U \subseteq G$ for $x \in G$. Thus, $\{[Hx]\}_{x \in G}$ partitions $G$ into disjoint subsets. In the language of group theory, $[Hx]$ can also be viewed as the union of the cosets in the orbit of $Hx$ under the conjugation action of $U$ on $G/H$.

It remains to define the edges of $\tilde{\mathcal{G}}$. For purposes of computation, the edge set of $\tilde{\mathcal{G}}$ is defined to be only a subset of the naturally induced edge set from $\mathcal{G}_H$. One would normally expect that for $[Hx] \neq [Hy]$, there is an edge between $[Hx]$ and $[Hy]$ if there is an edge in $\mathcal{G}$ between $x'$ and $y'$ for $x' \in [Hx]$ and $y' \in [Hy]$. For purposes of computation, it is more convenient to first choose an arbitrary transversal $T \subseteq G$ consisting of exactly one point from each distinct set $[Hx]$ for $x \in G$. Denote by $\overline{x}$ the unique element in $[Hx] \cap T$. We define $\tilde{\mathcal{G}}$ to have a directed edge between $[Hx]$ and $[Hy]$ labelled by $s \in S$ if and only if $[H\overline{x}s] = [Hy]$.

We next show that the edges in the two views of $\tilde{\mathcal{G}}$ really do correspond. There is an edge in the unlabelled graph if and only if there is at least one directed, labelled edge in the directed, labelled multigraph. The backward implication is easy. For the forward implication, suppose there is an edge in the unlabelled graph. So, there are $x' \in [H\overline{x}]$, $y' \in [Hy]$ and $s' \in S$ such that $x's' = y'$. Then there is a $u \in U$, such that $x'^u \in H\overline{x}$ and so $x'^u s'^u = y'^u \in [Hy]$. Note also that when $S$ is closed under inverses, if there is a directed edge from $[Hx]$ to $[Hy]$, then there is at least one directed edge from $[Hy]$ to $[Hx]$.

Suppose now that $g$ is an arbitrary element of $G$, and we want to construct a shortest word on $S$ whose action on $\mathcal{C}$ is the same as the action of $g$ on $\mathcal{C}$. This is the same problem as finding a shortest sequence, $g_1, \ldots, g_k$ of elements of $S$ such that $g(g_1 \cdots g_k)^{-1} \in H$, or equivalently, that $g \in Hg_1 \cdots g_k$. The next result shows how to construct such a sequence from any path of length $k$ from $[H]$ to $[Hg]$.

**Theorem 2.** *Let $g$ be an arbitrary element of $G$. There exists a path in $\tilde{\mathcal{G}}$ from $[H]$ to $[Hg]$ of length $k$ if and only if there exist $g'_1, \ldots, g'_k \in S$ such that $g \in Hg'_1 \cdots g'_k$.*

*Proof.* Suppose first that there exists a path of length $k$ from $[H]$ to $[Hg]$ in $\tilde{\mathcal{G}}$. Then there exists $g_1, \ldots, g_k \in S$ and $w_0, \ldots, w_k \in T$ (hence $w_i = \overline{w_i}$) such that $w_0 = 1$, $w_k = \overline{g}$ and $[Hw_{i-1}g_i] = [Hw_i]$. Note that $[Hw_{i-1}g_i] = [Hw_i]$ implies that there are $h_i \in H$ and $u_i \in U$ such that $w_i = h_i w_{i-1}^{u_i} g_i^{u_i}$. Iterating this, shows that $w_k = h g_1^{u'_1} \cdots g_k^{u'_k}$ for $h = h_k(h_{k-1}(h_{k-2}\cdots)^{u_{k-2}})^{u_{k-1}} \in H$ and $u'_i = u_i \cdots u_k$. Letting $g = h_{k+1} w_k^{u_{k+1}}$ and setting $g'_i = g_i^{u'_i u_{k+1}} \in S$ proves the assertion in the forward direction.

Conversely, suppose $g \in Hg'_1 \cdots g'_k$. Define $u'_i$ and $h'_i$ so that $w_i = h'_i (g'_1 \cdots g'_i)^{u'_i}$ satisfies $\overline{w_i} = w_i$. Let $w_0 = 1$. Note that $w_k = \overline{g}$ and $[Hw_i (g'_{i+1})^{u'_{i+1}}] = [Hw_{i+1}]$. This then gives the required path of length $k$ from $[H]$ to $[Hg]$ in $\mathcal{G}$. □

**3.1. Encoding of nodes of reduced graph.** Next, we consider how to produce a 1-1 mapping of the nodes of $\mathcal{G}$ into a "small" set of integers. The encoding serves as an almost perfect hash function of $\mathcal{G}$, in that the number of integers is not much larger than the number the number of nodes. The results in [8] show how to use such a 1-1 mapping to store a spanning tree using storage of only $\log_2(3)$ bits times the size of the "small" integer set. This is especially important for the larger graphs. For example, given 16 chips and 4 bus lines ($|\mathcal{C}| = 16$, $|\mathcal{B}| = 4$), the number of cosets, $Hg$, to explore is $|G/H| = (16+4)!/4! \approx 1.0 \times 10^{17}$. Yet the reduced graph has only 3,950 nodes. The following technique allows a unique encoding of those 3,950 nodes within the range [0, 5,822].

In order to construct a unique encoding of $[Hg]$, we require two auxiliary functions, $r(x, y)$ and $s(x, y)$. We denote by $r(x, y)$ the number of partitions of $x$ indistinguishable elements, such that the largest set within the partition is of size no larger than $y$. So, for example, $r(5, 2) = 3$ since the set of 5 elements can be partitioned into subsets of size $(2, 2, 1)$, $(2, 1, 1, 1)$ and $(1, 1, 1, 1, 1)$ for which no subset has more than 2 elements. In general,

$$r(x, y) = \sum_{i=1}^{y} r(x - i, \min(i, y, x - i)).$$

Also, $s(x, y)$ is the number of ways in which $x+y$ indistinguishable elements can be partitioned, where the $x + y$ elements are first split into two distinguished subsets, the first of size at most $x$ and the second of size at least $y$. So, for example, $s(1, 2) = 5$ and the partitions are of the form $((1), (2))$, $((1), (1, 1))$, $((), (3))$, $((), (2, 1))$ and $((), (1, 1, 1))$. In general,

$$s(x, y) = r(x, x)r(y, y) + s(x - 1, y + 1)$$

for $x > 0$ and $s(0, y) = r(y, y)$.

We now give a unique characterization of a node of the reduced graph in terms of a sequence of invariants. The proof that this is a unique characterization is easy, but is not included here. In the restriction of $g$ to the union of the cycles that fix $\mathcal{B}$, let $c_i$ be the length of the $i$-th such cycle. Let $b_i$ be the number of points $x \in \mathcal{C}$ such that $x^{g^j} \in \mathcal{C}$ for all $j < i$ and $x^{g^i} \notin \mathcal{C}$. Then $[Hg]$ is uniquely characterized by (a) the sequence $c_1, \ldots, c_{|\mathcal{C}|}$ and (b) the sequence $b_1, \ldots, b_{|\mathcal{B}|}$. Alternatively, one can

say that the $c_i$ and $b_i$ are invariants of any permutation $h \in [Hg]$ and the collection of invariants identifies a unique $[Hg]$.

We can now describe the encoding of the equivalence class $[Hg]$ in terms of its invariants, $c_i$ and $b_i$ for fixed $|\mathcal{C}|$. Let $b = \sum_i b_i$ and let $c = \sum_i c_i$. Note that $b + c = |\mathcal{C}|$. Then the encoding is given by

$$\text{Encoding of } [Hg]\text{: } s(b-1, c+1) + t((b_1, \ldots, b_{|\mathcal{B}|}))r(c,c) + t((c_1, \ldots, c_{|\mathcal{C}|})).$$

(We set $s(b-1, c+1) = 0$ when $b = 0$.) Here, for fixed $b$, if $\sum_{j=1}^{i} x_i = b$, then $t((x_1, \ldots, x_i))$ is a unique encoding of the partition $(x_1, \ldots, x_i)$ among all partitions of $b$ elements. The value of the function "$t$" is defined to be independent of the order of its arguments. We re-order the arguments so that $x_1 \geq \cdots \geq x_i$ and define

$$t((x_1, \ldots, x_i)) = t((x_2, \ldots, x_i)) + r(\sum_{j=1}^{i} x_i, x_1 - 1)$$

for $i > 0$ and $t(()) = 0$ for $i = 0$. Note that $t((x_1, \ldots, x_i)) \leq r(\sum_{j=1}^{i} x_i, \max_{1 \leq j \leq i} x_i) \leq r(\sum_{j=1}^{i} x_i, \sum_{j=1}^{i} x_i)$.

**3.2. Optimization: Use of Reduced Generating Sets.** Since the reduced labelled multigraph (defined in section 3) is a multigraph, an edge between nodes in the reduced graph may have multiple labels. One can recognize, a priori, that certain edge labels in the reduced graph are redundant. This allows us to moderate the exponential explosion in the number of generators as $|\mathcal{C}|$ and $|\mathcal{B}|$ increase. This is critical, since $|S| = \sum_{i=1}^{|\mathcal{B}|} \binom{|\mathcal{C}|}{i} \binom{|\mathcal{B}|}{i} i!$. To take an extreme example, for 16 chips and 4 bus lines, $|S| = 58{,}624$. Yet for $x = 1 \in G$, there are only 4 non-redundant generators, $g$, (each uniquely characterized by the number of transpositions in $g$) yielding distinct equivalence classes $[Hxg]$. All other elements of $xS$ are conjugate to one of the 12 under $U$.

In order to discover these redundant edge labels, consider the the node $[Hx]$, $\overline{x} = x$, with outgoing edge labelled by $g_0$. One must eliminate those $g \in S$ such that $g \neq g_0$ and $[H\overline{x}g] = [H\overline{x}g_0]$. Equivalently, for a given $x \in G$ and $g_0 \in S$, one must find those $g \in S$ with $g \neq g_0$ such that $h(\overline{x}g)^u = \overline{x}g_0$ for some $h \in H$ and $u \in U$.

The following three heuristics are used to remove certain $g \in S$ from consideration. The statement assumes some arbitrary ordering of the points $\mathcal{C} \cup \mathcal{B}$, so that "smallest" and "$<$" are well-defined. The correctness of these heuristics follows from the results of section 4, which define a total ordering on $S$. So, if $g, g_0 \in S$, $g_0 < g$, and the heuristic can establish that $[H\overline{x}g] = [H\overline{x}g_0]$, then $g$ is eliminated. Let $\mathcal{C}' = \{a\colon\ a^{x^i} \in \mathcal{C}\ \forall i\} \subseteq \mathcal{C}$.

(1) First, $g$ can be removed if there is a point $a \in \mathcal{C}'$ moved by $g$ that is not the smallest in the cycle of $\overline{x}$ containing $a$ and for which $g$ does not move the point that is first in that cycle. (There is a conjugation in $U$ that "rotates" the points of the cycle until $a$ is first in the cycle, while leaving $\overline{x}$ unchanged.)
(2) Second, $g$ can be removed when there is a point of $\mathcal{C}'$ moved by $g$ such that it is smallest in a cycle of $\overline{x}$ if there is an earlier cycle of points of the same length as the current cycle, and $g$ does does not move the point that

| # chips | # bus lines | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 4 | 6/12<br>0.05 | 3/17<br>0.2 | 3/19<br>0.7 | 3/20<br>0.7 | | | | |
| 5 | 7/19<br>0.1 | 4/28<br>0.3 | 3/33<br>1.0 | 3/35<br>3.1 | 3/36<br>5.6 | | | |
| 6 | 9/30<br>0.2 | 5/47<br>0.6 | 4/57<br>3.5 | 3/62<br>11.8 | 3/64<br>30 | 3/65<br>56 | | |
| 7 | 10/45<br>0.9 | 6/73<br>1.2 | 4/92<br>6.7 | 3/102<br>37 | 3/107<br>120 | 3/109<br>293 | 3/110<br>699 | |
| 8 | 12/67<br>0.4 | 6/114<br>2.2 | 5/147<br>17 | 4/167<br>104 | 3/177<br>460 | 3/182<br>1432 | 3/184<br>4259 | 3/185<br>13351 |

is smallest in that cycle. (In that case, there is a conjugation in $U$ that interchanges the two cycles, and so leaves $\overline{x}$ unchanged.)

(3) Third, $g$ can be removed if there are points $a, a' \in \mathcal{B}$ such that $a' < a$, $(a')^{\overline{x}} = a'$, $(a')^g = a'$, $a^{\overline{x}} = a$ and $a^g \neq a$. (The transposition $(a\ a') \in U$ leaves $\overline{x}$ unchanged.)

**3.3. Experimental Results.** The following table describes the maximum length permutation route, the CPU time, and the number of nodes in the reduced graph for various values of the number of chips and the number of buses. The computation was carried out under AKCL 1.615 (Common LISP). A SPARCstation-2 was used. For each tested combination of number of chips and bus lines, three numbers are reported. The first is the number of steps for an optimal permutation route for the worst case permutation (the diameter of the reduced labelled multigraph). The second is the number of nodes required to store the internal data structure. For large numbers of nodes, this will usually dominate the space requirements. Only two bits are required to store each node [8]. So, storage requirements are minimal for the examples shown. The third number is the number of seconds of CPU time to generate the data structure. From this data structure, an optimal permutation route can be computed in less than a millisecond for any desired permutation.

For 12 chips with 4 bus lines and 16 chips with 4 bus lines, two special tests were made. The results were a diameter of 5 with 919 nodes (5/919) and a diameter of 6 with 3,950 nodes (6/3950), respectively. The computation was carried out on a SPARCserver 670 in 43 minutes and 10 hours of CPU time, respectively.

The range of encodings is larger than, but usually close to the actual number of states. The range of encodings is independent of the number of bus lines. For example, at the beginning of section 3.1, we saw an example of 16 chips and 4 bus lines, in which the range [0, 5,822] was needed for 167 nodes. If the 2-bit encoding scheme is used, space must be allotted for the total range of encodings. Where the actual number of states used is much smaller, one can use hashing techniques to require space proportional to the actual number of states. Naturally, the constant of proportionality would be much higher than two bits.

Note that the length of the optimal permutation route for the worst case is always at least three. The reason for this is easy to see, since a one-step permutation route is required to move the data from the chips to the bus lines. A second one-step

permutation route moves the data from the bus lines to other chips. A third one-step route then places some of the data back on the bus lines. In practice, one is only interested in permutation routes that leave all data on the chip, and none on the bus lines. Hence, for permutation routes of interest, with sufficiently many bus lines, there is always a permutation route of length at most two.

## 4. Pruning Locally Redundant Generators

Although $\tilde{\mathcal{G}}$ was previously defined as the reduced labelled multigraph, we also identify it with the labels the nodes of the graph, which are the equivalence classes $[Hx]$ for $x \in G$. For $[Hx] \in \tilde{\mathcal{G}}$ and $y \in S$, we define $[Hx]y = [H(\overline{x}y)]$ as in section 3.

The next theorem characterizes a set of representatives, $\overline{x}S_x$, for the distinct equivalence classes $[Hx]S \subseteq \tilde{\mathcal{G}}$. Equivalently, this characterizes representatives for the nearest neighbors in the reduced labelled multigraph.

**Theorem 3.** *Let $\rho$ be a total ordering on $S$. Let $H \subseteq U$, $H^U = H$ and $S^U = S$. For $x \in G$, let $S_x = \{g \in S\colon\ g \leq_\rho h\ \forall h \in (\overline{x}^{-1}(\overline{x}g)^U H) \cap S\}$. Then $[Hx]S = [Hx]S_x$ and $|[Hx]S| = |S_x|$ — i.e. the number of nearest neighbors of $[Hx]$ in $\tilde{\mathcal{G}}$ is $|S_x|$.*

*Proof.* $[Hx]S = [Hx]S_x$ follows from the definitions of $[Hx]$ and $S_x$, along with the properties of $H$. Let $g, h \in S$. Then

$$\begin{aligned} [Hx]h = [Hx]g \quad &\Leftrightarrow \quad H(\overline{x}h)^U = H(\overline{x}g)^U \Leftrightarrow (\overline{x}h) \in H(\overline{x}g)^U \\ &\Leftrightarrow \quad h \in \overline{x}^{-1}H(\overline{x}g)^U = \overline{x}^{-1}(\overline{x}g)^U H. \end{aligned}$$

Thus if $g, h \in S_x$ and $[Hx]h = [Hx]g$ then $g$ and $h$ must both be lexically least, and so $g = h$. □

Define the *centralizer* of $x \in G$ in $U$ to be $\mathrm{Cent}_U(x) = \{g \in U\colon\ x^g = x\}$.

**Corollary 4.** *Let $\rho$ be a total ordering on $S$. Let $H \subseteq U$, $H^U = H$ and $S^U = S$. For $x \in G$, let $S'_x = \{g \in S\colon\ g \leq_\rho h\ \forall h \in (g^{\mathrm{Cent}_U(\overline{x})}H) \cap S\}$. Then $[Hx]S = [Hx]S'_x$ and $|[Hx]S| \leq |S'_x|$ for $[Hx]S \subseteq \tilde{\mathcal{G}}$.*

*Proof.* Note that since $\mathrm{Cent}_U(\overline{x}) \subseteq U$, $g^{\mathrm{Cent}_U(\overline{x})}H = \overline{x}^{-1}(\overline{x}g)^{\mathrm{Cent}_U(\overline{x})}H \subseteq \overline{x}^{-1}(\overline{x}g)^U H$. So, $S_x \subseteq S'_x \subseteq S$. □

The advantage of $S'_x$ over $S_x$ is that, given $\mathrm{Cent}_U(\overline{x})$, there are well-known techniques for enumerating the elements of $S'_x$ [4]. For $U$ equal to $\mathrm{Sym}(\mathcal{C}) \times \mathrm{Sym}(\mathcal{B})$, $\mathrm{Sym}(\Omega)$ and many other special cases of interest, computing $\mathrm{Cent}_U(\overline{x})$ is also efficient [10].

The heuristics of section 3.2 follow from the special case for $S'_x$. The order, $\rho$, is a variation of lexical ordering.

$$g >_\rho h \text{ iff } i^g = i^h\ \forall i < j, j^g \neq j^h, \text{ and either } j^h = j \text{ or } j^g < j^h \neq j$$

The use of a lexical ordering means that one can generate all elements of $S'_x$ in lexical order and if a permutation $g \in X'_x$ satisfies $i^g \leq i^h$ or $i^h = i\ \forall i \leq j$, $\forall h \in g^{\mathrm{Cent}_U(\overline{x})}H$, then the subtree $\{h \in g^{\mathrm{Cent}_U(\overline{x})}H\colon\ j^g \neq j^h, i^g = i^h\ \forall i < j\} \cap S'_x = \emptyset$ and so the entire subtree can be eliminated.

Where it is feasible to compute $S_x$ or where $S'_x$ is not too much larger than $S_x$, the entire computation of a data structure for the spanning tree of the reduced labelled multigraph has a nice complexity characterization. Since $|S_x| \leq |\tilde{\mathcal{G}}$, the algorithm examines at most $|\tilde{\mathcal{G}}|^2$ directed edges in its breadth-first search. Note that enumeration of $S'_x$ in lexical order costs $O(|S'_x|n)$ for $n = |\Omega|$, since on the current branch, either the entire subtree will be eliminated, or a new element of $S'_x$ will be found before the necessity of backtracking. Computing the product $xg$ and its encoding cost $O(n)$. So, the total cost of the algorithm is $O(|\tilde{\mathcal{G}}|^2(n+E) + |\tilde{\mathcal{G}}|(A + Rn))$, where $A$ is the cost of computing the automorphism group and $R$ is the cost of enumerating the elements of $S_x$ of $S'_x$. Further, for the bus interconnection networks of section 3, $O(|\tilde{\mathcal{G}}|^2 n)$ is the dominant term.

## 5. Generalization to Other Models

The previous section was described under the assumption that each chip had only one register (and hence hold only one piece of data at a time). Naturally, most implementations will allow multiple registers or memory cells on a single chip. Hence, shortest paths derived from previous calculations serve only as an upper bound on the shortest length path when multiple registers are allowed.

The model can be extended to $k$ registers per chip, by including $k$ copies of each chip. Every generator must also be replicated to include variation in which each bus is connected to an arbitrary register. Finally, the subgroup $H$ must be extended to include generating elements that transpose arbitrary registers from the same chip.

The previous section was described in a situation with full connections between chips and registers. If there are fewer connections, in general, one will need to choose a subgroup of $U = \mathrm{Sym}(\mathcal{C}) \times \mathrm{Sym}(\mathcal{B})$ that preserves the architecture under conjugation. Thus, the subgroup $U$ can be viewed as the automorphism group of a labelled graph, where each node of the graph is labelled either by $\mathcal{C}$ or by $\mathcal{B}$. Usually ease of manufacturing dictates a uniform architecture, and so most real-world example are still likely to have a fairly large automorphism group.

Finally, a few remarks about generalizations to other models besides bus interconnection networks is worthwhile. There is a common difficulty in translating routes in physical networks to permutation routes. Suppose one has nodes $A$, $B$ and $C$ with edges between $A$ and $B$, and between $B$ and $C$. One would like to add a transposition between $A$ and $B$ as a one-step permutation route. Similarly, one would like to add a transposition between $B$ and $C$. Finally, one would like to to move data from $A$ to $B$ while other data is moved from $B$ to $C$, in parallel, in a single one-step permutation route. However, a naive product of transpositions would send data from $A$ to $C$, which cannot be done in one step. The solution is to split each node into two nodes corresponding to two registers/ports: a sending port and a receiving port. Thus, one can move data from $A$ to $B$ in parallel with distinct data moving from $B$ to $C$, and this can be expressed as a permutation.

## 6. Conclusion

It has been demonstrated how to find permutation routes for larger data structures than would be possible by a direct approach. The solution involves pre-computing a special data structure in seconds. One can then derive shortest permutation routes achieving an arbitrary permutation from the pre-computed data

structure. Finding that solution is a computation that can be carried out in less than a millisecond (and much shorter times are possible with hardware support).

In implementations, one would pre-compute the data structure for a given hardware architecture off-line. The required space is very small, although the CPU time for the pre-computation can be appreciable. Since the pre-computation is done once only, long CPU times may be acceptable. Specific permutation routes would either be computed at compile-time for an individual application, or else it would be be computed as part of a set-up routine for a route determined at run-time. In the latter case, one would expect the computed permutation route to be used many times to justify the overhead of computing such a route at run-time.

## 7. Acknowledgements

The authors wish to thank Richard Draper, Chuck Fiduccia and Bryant York for stimulating discussions.

## References

1. F. Annexstein and M. Baumslag, "A Unified Framework for Off-line Permutation Routing in Parallel Networks", Math. Syst. Theory **23** (1991), pp. 233–252.
2. F. Annexstein, M. Baumslag, and A.L. Rosenberg, "Group Action Graphs and Parallel Architectures", SIAM J. Computing **19** (1990), pp. 544–569.
3. J-C. Bermond, C. Delome, and J-J. Quisquater, "Strategies for interconnection networks: Some methods from graph theory", *Journal of Parallel and Distributed Computing* **3** (1986), pp. 433-449.
4. G. Butler, *Fundamental Algorithms for Permutation Groups*, Lecture notes in computer science **559**, Springer-Verlag, New York (1991).
5. L. Campbell, G.E. Carlsson, and M.J. Dinneen, "Small Diameter Symmetric Networks from Linear Groups", *IEEE Transactions on Computers* **41**, no. 2, (1992), pp. 218–220.
6. G.E. Carlson, J.E. Cruthirds, H.B. Sexton, and C.G. Wright, "Interconnection networks based on a generalization of cube-connected cycles", *I.E.E.E. Trans. Comp.* **C-34** (1985), pp. 769–777.
7. D.V Chudnovsky, G.V. Chudnovsky and M.M. Denneau, "Regular Graphs with Small Diameter as Models for Interconnection Networks", *I.E.E.E. Trans. Comp.* (1988), pp. 232–239.
8. G. Cooperman and L. Finkelstein, "New Methods for Using Cayley Graphs in Interconnection Networks", *Discrete Applied Mathematics* **37/38** (special issue on Interconnection Networks) (1992), pp. 95–118.
9. G. Cooperman, L. Finkelstein and N. Sarawagi, "Applications of Cayley Graphs", *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC-8, Tokyo, 1990)* Springer-Verlag Lecture Notes in Computer Science **508**, S. Sakata (ed.) (1991), pp. 367–378.
10. E. Luks, Permutation Groups and Polynomial-Time Computation, *Proceedings of DIMACS Workshop on Groups and Computation*, DIMACS-AMS **11**, L. Finkelstein and W.M. Kantor (eds.), AMS, Providence, RI, (1993), pp. 139–175.
11. M. Ramras, "Routing Permutations on a Graph", *Networks* **23**, no. 4, (1993), pp. 391–398.
12. S.T. Schibell and R.M. Stafford, "Processor Interconnection Networks from Cayley Graphs", Disc. Appl. Math. **40** (1992), pp. 333–362.
13. C.C. Sims, "Computation with Permutation Groups", in *Proc. Second Symposium on Symbolic and Algebraic Manipulation*, S.R. Petrick (ed.), ACM, New York (1971).

College of Computer Science, Northeastern University, Boston, MA 02115
*E-mail address*: gene@ccs.neu.edu

College of Computer Science, Northeastern University, Boston, MA 02115
*E-mail address*: laf@ccs.neu.edu