The Matching Model for Routing Permutations on Graphs

A Senior Project Submitted to
the Division of Natural Sciences and Mathematics
of Bard College

by Rodney L. Waldhoff

Annandale-on-Hudson, New York
May 1997

Abstract

A model for routing on connected graphs is investigated. On each vertex of an undirected, connected graph, $G = (V, E)$, is placed a unique "pebble". Each pebble has a destination which a unique vertex in V, so that the map from a vertex v to the destination of the pebble on v forms a permutation of the vertex set. At each step in the routing, a matching of the edges of G is selected, and the pebbles on the pair of vertices incident with each edge in the matching are interchanged. Our goal is to route all pebbles to their destinations in a minimal number of steps and we define the routing number of a graph, $\mathrm{rt}(G)$, to be the maximum number of steps ever required to route a permutation on G with a minimal length routing. The matching model is compared to parallel sorting algorithms and an algorithm for routing on cycles with an even number of vertices is presented. This algorithm is used to determine the routing number for even cycles. Bounds on the routing number for several classes of graphs are determined. Finally, a new lower bound for any graph based upon diameter is considered.

Thanks are due to many:

to my family, for unwavering support.

to my fiancée, Stephanie, for putting up with me the past few months and for agreeing to marry me nonetheless.

to my advisor, Mark Halsey, for keeping faith, for having the patience to listen to my ramblings until they actually made sense and for guidance along the way.

to Richard Goldstone for sharing ideas, abstract nonsense and TEX-nical advice.

to Ethan Bloch, for teaching me the meaning of the word "rigor" in mathematics.

to Michael Lewis for various motivations and illuminations.

and to everyone else I tormented for advice, opinions or support in the process of completing my Senior Project.

There is nothing more difficult to take in hand, more perilous to conduct, or more uncertain in its success, than to take the lead in the introduction of a new order of things.

—Niccol Machiavelli, The Prince

Contents

1. Motivations

The foundations of modern computer science are typically traced to two names, Alan Turing and John Von Neumann.

Turing is credited with the concept of the Turing Machine, a theoretical model for any deterministic mind. In a Turing Machine, we imagine a processor moving along an infinite tape or "memory". The tape is divided into sections, and a symbol may be written on each section. At any given point in time, the processor may change its position on the tape, or read or change the symbol at its current position. Which action is to be performed is determined by which *state* the processor is currently in, and the processor's state may change in response to the symbols it reads. Turing's model was created in order to explore issues of decidablity and computability, but became the basis of a practical design for general purpose computing machines, as developed by Von Neumann.

Although the Von Neumann architecture is slightly more complex, the essence of the Turing Machine persists. Rather than a moving processor, we have a "data-bus" which carries information from the memory source to a central processing unit, instructing it to perform a sequence of operations that manipulate the information stored in memory. More than 50 years after it was introduced, computers are still built according to Von Neumann's basic design.

Yet there is limit to the power of this single processor/single memory architecture, in that only one instruction may be performed at a time. Great increases in speed could be achieved if it were possible to perform independent actions in *parallel.* One obvious solution to this problem is simply to use multiple processors, each sharing the single memory source. This introduces a new problem, concurrency—when two processors wish to access the same section of memory, or even worse, when one processor wants to change the memory that another is currently reading.

To get around the problem of concurrency, the computer industry has began to move toward *distributed computing*, whereby many processors, each with its own memory source, operate independently, communicating results to one another when necessary. This is the foundation for many of the so-called *massively parallel* architectures. In such systems the organization of the communication links between processors becomes pivotal, in that they effect not only the cost and size of a given system, but also the nature of the algorithms that can be performed upon them efficiently.

Hence a primary task in the development of parallel computing systems is to determine which structures are best suited for parallel architectures. In all likelihood the answers lie in the area of mathematics known as graph theory. This paper considers a graph-theoretic model introduced by researchers at Bell Laboratories in order to explore this issue and others, and you'll find (I hope) that this model raises some interesting mathematical questions.

2. A Brief Introduction to Graph Theoretic Terminology

Graph theory provides a framework for talking and reasoning about interconnections between a collection of objects, and in this section we lay the foundation for that framework. It is unfortunate that the term *graph*, as it is used here, has quite a different meaning than what you might be familiar with from analytic geometry or real analysis, where a graph is used to plot the change in one variable versus change in another. In discrete mathematics, a graph is similar to what is sometimes known, particularly in computer science circles, as a *network*.

Intuitively, a graph is an abstract representation of a collection of "places" and of "roads" that describe how we might get from one place to another.

Definition 2.1. A *graph* $G = (V, E)$ is a collection of two sets, a (non-empty) set of *vertices*, V, and a set of *edges*, E. Each edge $e \in E$ is associated with one or two vertices in V. Sometimes we denote the vertex set of a graph G by $V(G)$ and the edge set by $E(G)$.

To define a graph, we must specify each of these sets, for example

$$G_1 = (\{v_1, v_2, v_3, v_4, v_5\}, \{e_1, e_2, e_3, e_4, e_5, e_6\})$$

where the edges are defined by

edge	associated vertices
e_1	$\{v_1, v_2\}$
e_2	$\{v_2, v_3\}$
e_3	$\{v_3, v_4\}$
e_4	$\{v_1, v_4\}$
e_5	$\{v_1\}$
e_6	$\{v_3, v_4\}$

.

It is often more convenient to present a graph pictorially, where each vertex is represented as a point and each edge as a line between the two points associated with it. Figure 1 shows three representations of the graph G_1.

A large vocabulary has been built up to describe graph theoretic concepts and only a small part of it is introduced here. Although there are several terms defined below, I think you'll

FIGURE 1. Each of these diagrams represent the same graph.

find that most of them mean more or less what your intuition suggests. This is, however, both a blessing and a curse. Due to the ambiguity of the 'non-mathematical' meanings of these terms, some authors define them slightly differently and some use different words for the same concept. Bearing this in mind, you may wish to skim over these definitions even if you are already familiar with graph theory.

Definition 2.2. The vertices in a graph that are associated with a particular edge are said to be *incident with* or *on* that edge. Two vertices that are incident with the same edge are *adjacent* or *co-incident.*

Definition 2.3. The *degree of a vertex* v, often denoted $\deg(v)$, is the number of edges it is incident with.

Definition 2.4. A graph G for which every vertex has the same degree is said to be *regular*. In particular, if $\deg(v) = k$ for all $v \in V(G)$ then G is said to be *regular of degree* k.

Definition 2.5. An edge is said to *connect* or *join* the vertices that are incident with it. The vertices joined by an edge are called the *endpoints* of that edge.

Definition 2.6. Two distinct edges that join the same vertices are *parallel*, and an edge that has exactly one endpoint is called a *loop*.

Definition 2.7. A graph that contains no parallel edges or loops is called a *simple graph.*

When a graph is simple, it is common to represent the edges by the vertices they connect, such as $\{v, w\}$ for the edge that joins v and w. Since simple graphs will be the focus of this paper, we will follow that convention here.

Definition 2.8. Let $G = (V, E)$ be a graph. A subset of edges, $m \subseteq E$, is said to be *vertex-disjoint* if no two edges in m have any endpoints in common ($e \cap f = \emptyset$ for all e, $f \in m$). A vertex-disjoint collection of edges from G defines a *matching* on G.

Definition 2.9. A *walk* from a vertex v_0 to a vertex v_n is an alternating sequence of vertices and edges,

$$\langle v_0, e_1, v_1, e_2, \ldots v_{n-1}, e_n, v_n \rangle$$

where each v_i is a vertex and each e_i is an edge joining v_{i-1} and v_i. A *path* is a walk that does not repeat an edge. (In symbols, $e_i \neq e_j$ for all $i \neq j$.) If the first and last vertices in a path are the same (i.e., $v_0 = v_n$) then the path is called a *cycle.* A *simple path* is a path that does not repeat a vertex ($v_i \neq v_j$ for all $i \neq j$) and a *simple cycle* is a cycle for which only the first and last vertices are the same ($v_i \neq v_j$ for all $0 < i < n$ and $0 < j < n$ with $i \neq j$ while $v_0 = v_n$).

Definition 2.10. The *length* of a walk, path, or cycle is the number of edges in the sequence.

When a graph is simple, it is often easier to represent a walk, path, or cycle by just listing the edges in the sequence, since this uniquely determines the vertices. We will follow that convention here.

Definition 2.11. The *shortest path from v to w* is, as the name implies, a path such that every other path from v to w is longer. We would like to refer to "a shortest path" when no other path is shorter, but this is not grammatically accurate, so we shall refer to these paths as *minimally short.* The *distance* between two vertices v and w in a graph G, denoted $\mathrm{d}\,(v, w)$, is the length of a minimally short path from v to w.

Definition 2.12. The *diameter* of a graph G, denoted $\mathrm{diam}(G)$, is the length of the longest minimally short path between any two vertices, i.e.,

$$\mathrm{diam}(G) = \max\left\{\mathrm{d}\,(v, w) : v, w \in V(G)\right\}.$$

Definition 2.13. If H is a graph such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$ for some graph G, then we say H is a *subgraph* of G, and denote this by $H \subseteq G$. A subgraph H is a *vertex-covering subgraph* or simply a *vertex-cover* if $V(H) = V(G)$. Given a subset of vertices $S \subseteq V(G)$, the subgraph formed by S together with all edges in $E(G)$ whose endpoints are in S is known as the *subgraph of G induced by S*. Such graphs are known as *induced subgraphs*.

Definition 2.14. A graph is said to be *connected* if there exists a walk from each vertex to every other vertex. A graph that is not connected is *disconnected*.

All the graphs we consider in this paper will be connected.

Definition 2.15. If there exists a bijective function $f : V(G) \to V(H)$ between two graphs G and H such that $\{f(v), f(w)\}$ is an edge in H if and only if $\{v, w\}$ is an edge in G (for any vertices $v, w \in V(G)$), then we say *G is isomorphic to H* and denote this $G \simeq H$. The bijection f is called the *isomorphism*.

Intuitively, two graphs are isomorphic if they simply use different names for the same vertices. It is easy to verify that graph isomorphism is an equivalence relation. For most purposes, including ours, isomorphic graphs are essentially the same graph.

Definition 2.16. Given two graphs, G and H, we may form the *Cartesian product* of G and H, denoted $G \times H$, where the vertex set consists of ordered pairs of vertices from the original graphs, i.e.

$$V(G \times H) = \{g \times h : g \in V(G) \text{ and } h \in V(H)\}.$$

The edges of the product graph connect two vertices whenever we hold one vertex in the pair fixed and there is an edge in the 'factor' graph between the vertices that that vertex is paired with, i.e.,

$$\begin{aligned} E(G \times H) &= \{\{g_1 \times h, g_2 \times h\} : h \in V(H) \text{ and } \{g_1, g_2\} \in E(G)\} \\ &\quad \cup \{\{g \times h_1, g \times h_2\} : g \in V(G) \text{ and } \{h_1, h_2\} \in E(H)\}. \end{aligned}$$

Although the vertices of $G \times H$ are ordered pairs, we denote them in the form $a \times b$ rather than (a, b) in order to avoid confusion of vertices with edges. Note that $G \times H \simeq H \times G$ and that $G \times H$ has $|V(G)| \cdot |V(H)|$ vertices and $|E(G)| \cdot |V(H)| + |E(H)| \cdot |V(G)|$ edges.

I find it easiest to picture the construction of $G \times H$ as replacing each vertex in H with a 'copy' of G, and connecting the corresponding vertices in two copies of G if there is an edge in H between the vertices they replaced. An example of this construction is illustrated in Figure 2.

FIGURE 2. A Cartesian product graph.

3. The Matching Model

3.1. **A Model for Parallel Processing.** The matching model for routing permutations on undirected, connected graphs was first proposed by Alon, Chung and Graham in [1][2]. The model is defined as follows.

Informally, consider a simple, connected graph $G = (V, E)$, where V is the vertex set and E is the edge set. Each vertex $v \in V$ is occupied by a unique pebble. Each pebble has a unique destination $\pi(v) \in V$, so that π defines a permutation of the vertex set. We wish to route each pebble to its destination using the following method. At each step a vertex-disjoint collection of edges (a matching) is chosen, and for each edge in the matching, the pebbles on the endpoints of the edge are interchanged. We want to route all pebbles to their destinations as quickly as possible, and we define the routing number of a graph and a permutation, $\mathrm{rt}(G, \pi)$, to be the minimal number of time steps required. Similarly, the routing number of a graph, $\mathrm{rt}(G)$, is defined to be the maximal $\mathrm{rt}(G, \pi)$ for all π, the best we can do for a worst-case permutation.

The restriction that G be connected simply ensures that every permutation is possible (otherwise, there would be no way to move a pebble between the components of the graph). Allowing G to contain duplicate edges or loops would not effect the routing number, since exactly one edge between any pair of vertices can be included in a matching, and including a loop in a matching wouldn't "move" the pebble at all. So for the sake of simplicity will shall assume G is simple.

It is easy to see how the matching model might be used to evaluate the utility of parallel architectures. If we imagine that each pebble represents a message or datum to be communicated between processors and we assume that the time needed to operate upon a message is negligible, or at least independent of the time required to communicate results between processors, we see that the routing number of a graph provides a useful bound upon the time required to execute each stage of a parallel algorithm. Even if a slightly different communications model is used, the routing number provides a relatively simple measure of the feasibility of a given structure for parallel systems.

3.2. **The graph-theoretic construction.** Throughout this section, let $G = (V, E)$ be a simple, connected graph.

Definition 3.1. Let $\pi : V \to V$ be a permutation of the vertex set. We call $\pi(v)$ the *pebble* on the vertex v. By abusing our notation a bit, if $\pi(v) = w$ we say that w is *the destination of the pebble on the vertex v*. Similarly, we shall say that *the pebble whose destination is w* is on the vertex $\pi^{-1}(w)$. When necessary, we will denote this pebble as p_w.

Definition 3.2. A matching in G is a subset of E that is vertex-disjoint (see Definition 2.8). For any permutation $\pi : V \to V$, a matching m defines the permutation $m \circ \pi : V \to V$ where

$$m \circ \pi(v) = \begin{cases} \pi(w) & \text{if } \{v, w\} \in m \\ \pi(v) & \text{otherwise} \end{cases} .$$

Transforming π to $m \circ \pi$ is known as *applying* a matching to a permutation.

Note that $m \circ m \circ \pi = \pi$, so each matching is its own inverse.

Definition 3.3. A *routing* r is a sequence of (non-empty) matchings to be applied. The *time* required for a routing $r = \langle m_1, m_2, \ldots, m_n \rangle$ is the number of (not necessarily distinct) matchings in the sequence, in this case, n. *Applying* a routing to a permutation means to apply each matching in turn. Hence $r \circ \pi = m_n \circ m_{n-1} \ldots \circ m_1 \circ \pi$. A routing is said to *achieve* or *route* π on G if the application of r to π yields the identity permutation.

Note that if $r = \langle m_1, \ldots, m_n \rangle$ achieves π on G, then $r^{-1} = \langle m_n, \ldots, m_1 \rangle$ will transform the identity permutation into π, so we can consider a routing as either sorting or distributing the pebbles. Our goal is to achieve π on G with a minimal time routing.

Definition 3.4. We define the *routing number of a permutation on a graph G*, denoted $\mathrm{rt}(G, \pi)$ to be the time required by a minimal routing that transforms π into the identity permutation.

Definition 3.5. We define the *routing number of a graph G*, denoted $\mathrm{rt}(G)$, to be the maximum time required to route any permutation on G. Hence

$$\mathrm{rt}(G) = \max \{\mathrm{rt}(G, \pi) : \text{for all permutations } \pi : V \to V\} .$$

The following definitions, while not strictly necessary, will make some of our discussion easier. They correspond directly to terminology used in abstract algebra.

Definition 3.6. We say a vertex v and the pebble $\pi(v)$ are *fixed* if $\pi(v) = v$.

Definition 3.7. A matching is said to *move* the pebble on the vertex v if v is the endpoint of some edge in the matching.

The notation $r \circ \pi$ defined in Definition 3.3, becomes somewhat cumbersome when we wish to discuss the action of routing upon a permutation in terms of specific vertices. So, when the routing and permutation are clearly defined by context, we shall use the following conventions.

Notation 3.1. For a routing $r = \langle m_1, m_2, \ldots, m_n \rangle$, we shall use the notation r_k to refer to the routing composed of the first k matchings in r, for $1 \leq k \leq n$. Hence $r_k = \langle m_1, m_2, \ldots, m_k \rangle$. Note that r_j is not defined for $j > n$ and $j < 0$. Also, we have $r_0 \circ \pi = \pi$ for any routing and $r_n = r$ whenever the time required for r is n.

Notation 3.2. For a given routing r and a given permutation π, we shall use π_i to denote $r_i \circ \pi$. Hence $\pi_i(v) = r_i \circ \pi(v) = m_i \circ \ldots \circ m_2 \circ m_1 \circ \pi(v)$ for all $v \in V$.

3.3. **The algebraic construction.** We can, of course, present the problem in purely algebraic terms, although the restrictions upon the available transpositions then appear somewhat arbitrary.

Given a set V, and a collection E of two-element subsets of V, we seek to represent a permutation π of the elements of V as a composition of terms $m_1 \circ m_2 \circ \ldots \circ m_t$, where each m_n $(1 \leq n \leq t)$ is the composition of transpositions $(v_1, v_2) \circ (v_3, v_4) \circ \ldots \circ (v_{k-1}, v_k)$ such that $v_i \neq v_j$ whenever $i \neq j$ and each pair $\{v_i, v_{i+1}\}$ is an element of E. We call such sequences *routings*, and define the *time* required by a routing to be the number of terms that must be composed. Then the routing number of the graph G defined by V and E is the same as that given in Definitions 3.4 and 3.5.

Most of the results and arguments we will consider are based upon graph theoretic concepts, so we will have little use for this algebraic construction.

FIGURE 3. The Cayley digraph representation of all routings on P_3 and C_3.

3.4. **Another View.** A tempting representation of this routing problem is to build a larger graph that encapsulates all of the necessary information.

Given a graph G, we construct a new graph G^*. The vertices of G^* consist of all permutations of $V(G)$. We place an edge between two permutations $\pi_1, \pi_2 \in V(G^*)$ if there is a matching in G that transforms one permutation into the other. (Since every matching is its own inverse, we know that this relation is symmetric and hence the edges of G^* are well-defined.) Readers familiar with abstract algebra will recognize this as the Cayley digraph representation of the group of permutations of $V(G)$ with the set of matchings in G as generators. Figure 3 illustrates this construction for the two connected graphs on 3 vertices.

Since the edges in G^* represent matchings on G, a walk in G^* is a routing on G. Hence this representation provides a visual summary of all possible routings. Moreover, we have $\mathrm{rt}(G) = \mathrm{diam}(G^*)$, since a minimally short path in G^* corresponds to a minimal time routing of a permutation on G. This allows us to convert our routing problem into one that is superficially simpler. Unfortunately, G^* has $(|V(G)|)!$ vertices, so it becomes difficult to reason about, or even picture, the structure of G^* for large G. It is easy to see that G^* is always regular with a degree corresponding to the number of matchings in G.

4. Instructive Trivia

In [2] Alon, Chung and Graham provide a comprehensive introduction to the matching model. Some of their results are fairly trivial (once you've seen them), but are worth reviewing in order to get our bearings and to provide a foundation upon which to build.

The construction of our model makes it easy to provide a lower bound for routing on any graph based upon its diameter, as stated in the following proposition.

Proposition 4.1 (The Elusive Diameter Bound). *For any graph G we have* $\mathrm{rt}\,(G) \geq \mathrm{diam}\,(G)$. (This is Proposition 1 in [2].)

Proof. This follows immediately from the observation that a pebble can travel at most one edge per step, so a permutation that maps a pebble from one "end" of the diameter to the other requires at least $\mathrm{diam}\,(G)$ steps. ∎

We might as well add the following observation, which constructs a similar bound on specific permutations.

Corollary 4.2. $\mathrm{rt}\,(G, \pi) \geq \max \{\mathrm{d}\,(v, \pi\,(v)) : v \in V\,(G)\}$.

In [2] the authors comment that "it would be interesting, but probably difficult to characterize graphs for which equality holds [in Proposition 4.1]." This is a question that we will consider in later sections. It seems that such graphs are rather rare. For now, note that the graph on two vertices is a graph for which equality holds.

In the best case for the routing problem we may transpose any two pebbles. This corresponds to a graph with an edge between every pair of vertices, as illustrated in Figure 4. Such graphs are known as *complete*.

Definition 4.1. The graph with n vertices whose edge set consists of all $\frac{n(n-1)}{2}$ pairs of vertices is known as *the complete graph on n vertices* and is denoted K_n. Note that K_n is the unique graph on n vertices that is regular of degree $n-1$.

Proposition 4.3 (A Useless Characterization of K_n). *Let G be a graph with $|V\,(G)| \geq 3$. Then G is a complete graph iff* $\mathrm{rt}(G) = 2$.

FIGURE 4. The complete graphs K_3, K_4 and K_5.

Proof. In [2], it is shown that $\mathrm{rt}(K_n) \leq 2$ by observing that we may consider disjoint orbits of a permutation independently (since every induced subgraph of K_n is also a complete graph), and that any orbit $(v_0, v_1, v_2 \ldots v_k)$ can be achieved at most two steps by routings of the form

$$\langle \{\{v_0, v_k\}, \{v_1, v_{k-1}\}, \ldots\}, \{\{v_2, v_k\}, \{v_3, v_{k-1}\}, \ldots\} \rangle .$$

Moreover, for graphs with more than two vertices there are permutations that require both steps. If a permutation π contains an orbit consisting of three elements, say (a, b, c), then clearly no single matching will route all of these pebbles to their destinations, since matching one of these pebbles with its destination precludes fixing the pebble it is matched with. For example, suppose we have the edge $\{a, b\}$ in the first matching. Then $r_1(b) = b$, but $r_1(a) = c$, so we cannot simultaneously fix both a and b with one matching.

Hence we need only show that $\mathrm{rt}(G) = 2$ implies that G is a complete graph. We will show the contrapositive. Suppose G is not complete. Then there exists vertices u,$w \in V(G)$ with no edge between them. Then a permutation π that maps u to w and w to u, ($\pi(u) = w$ and $\pi(w) = u$) holding all other pebbles fixed, requires at least three steps, which we can see by simply following the action of one of the pebbles, say p_u.

In the first matching, we can move p_u to some intermediate vertex v, but this must displace the pebble p_v that was initially fixed. A second matching might move p_u to its destination, but we cannot simultaneously replace p_v, since its destination v is already matched with the vertex u. Hence a third matching is required to replace the pebble that was displaced. So if G is not complete it takes at least 3 steps to achieve this permutation. ∎

The complete graph on n vertices provides a lower bound for all graphs on n vertices, since removing edges from a graph will never decrease the routing number. (That is, $\mathrm{rt}(H) \geq \mathrm{rt}(G)$ for any vertex covering subgraph $H \subseteq G$.) Moreover, Proposition 4.3 not only demonstrates that complete graphs fail to meet the Elusive Diameter Bound (since $\mathrm{diam}(K_n) = 1$), but also that the same is true for graphs that are 'not quite complete':

Corollary 4.4. *If G is a graph with $\mathrm{diam}\,(G) = 2$, then $\mathrm{rt}\,(G) > 2$.*

The following theorem from [2] is far from trivial, but rather useful for bounding the routing number of a Cartesian product graph in terms its 'factors'.

Theorem 4.5 (The Remarkable Cartesian Product Bound)**.** *Let G and H be graphs. Then $\mathrm{rt}\,(G \times H) \leq \mathrm{rt}\,(G) + 2 \cdot \mathrm{rt}\,(H)$.* (This is Theorem 4 from [2].)

A full proof of this theorem would take us a bit far afield here, but let us note that the proof is based largely upon the rather surprising fact that in order to sort a two-dimensional array, we can sort the entire array by first sorting the elements in each row, then the elements in each column, and then sorting the rows again. (This result is proven in [3], among others.) If we think of each copy of H as a "row" of $G \times H$ and each copy of G as a "column", we begin to see the main thrust of the argument.

Note that it doesn't matter which graph we call G or which graph we call H, since $G \times H \simeq H \times G$, so we may rewrite Theorem 4.5 as

$$\mathrm{rt}\,(G \times H) \leq \max\left\{\mathrm{rt}\,(G)\,, \mathrm{rt}\,(H)\right\} + 2 \cdot \min\left\{\mathrm{rt}\,(G)\,, \mathrm{rt}\,(H)\right\}.$$

5. Paths and Cycles

In the study of parallel architectures, we find that often our intuition from the analysis of serial algorithms can be misleading. Consider the *bubble-sort* algorithm, an introductory technique for sorting an array of numbers. To bubble-sort an array, we first compare the first and second elements, exchanging them if they are out of order, then compare the second and third elements, and so on. When we reach the end of the array we repeat the process, until no exchanges are made on some pass through the array. I describe this algorithm as "introductory" since it is far too slow for real-world applications, with a running time proportional to the square of the number of elements to be sorted. Yet an obvious parallelization of this algorithm (obvious under the model we consider here, at least), described in Section 5.1.3, runs in linear time.

5.1. Comparator Networks.

In [3], Donald Knuth discusses some specific cases of the matching model in the guise of sorting in parallel with comparator networks. A comparator network allows for the implementation of a parallel version of comparison/exchange sorting techniques. The following section recasts Knuth's construction in the terms of our problem.

5.1.1. *A Comparator Network Construction.*

For a given graph G, we want to specify a sequence of matchings, $\langle M_1, M_2, \ldots M_m \rangle$, so that every permutation can be achieved by a routing whose matchings are subsets of the corresponding matching in the sequence. We will call such sequences *comparator networks*. In fact, we will build a framework that is capable of handling pebbles whose destinations are not fully specified, so that a comparator network will "sort-out" an order relation on a set of pebbles so that it is non-decreasing with respect to an order relation upon the vertex set $V(G)$.

- Let $<_V$ be a total order relation on the vertex set $V(G)$. (For every pair of vertices v, $w \in V(G)$ either $v <_V w$ or $w <_V v$.)

- Let ϕ be a map from $V(G)$ to a set of pebbles, P. Let $\leq_\phi$ be an order relation on P that may be symmetric. (So we may have $p \leq_\phi q$ and $q \leq_\phi p$ for some distinct $p, q \in P$.) Let us call ϕ a *pebbling* of the vertex set.
- A matching m on G defines a function $m \circ \phi : V(G) \to P$ in exactly the same way it transforms a permutation, as described in Definition 3.2.
- A sequence of potential matchings, $\langle M_1, M_2, \ldots M_m \rangle$, is determined off-line (before the initial pebbling is known). We say that the network defined by $\langle M_1, M_2, \ldots M_m \rangle$ has m stages.
- The actual routing, $r = \langle m_1, m_2, \ldots m_m \rangle$ is determined dynamically, according to the following rule: An edge $\{v, w\}$ with $v <_V w$ is included in the matching m_k if $\{v, w\} \in M_k$ and $r_{k-1} \circ \phi(w) \leq_\phi r_{k-1} \circ \phi(v)$ (for $1 \leq k \leq m$). Hence pebbles are exchanged when they are out of order.
- We wish to construct comparator networks that will route any ϕ such that $r \circ \phi(v) \leq_\phi r \circ \phi(w)$ whenever $v <_V w$. (Networks that 'sort' the pebbles so that the relation $\leq_\phi$ is monotonic with respect to the relation $<_V$.)

For most of our purposes, we may consider the P to be the vertex set, ϕ to be a permutation and let the relations $<_V$ and $\leq_\phi$ be the same. In that case, since each vertex and pebble has a unique position in their respective orderings, the smallest pebble will be sorted to the smallest vertex, the second smallest pebble to the second smallest vertex, and so on, so that a "sorted" pebbling is a "routed" pebbling. We need to allow for the possibility of non-distinct values, however, in order to apply Proposition 5.1.

Proposition 5.1, which Knuth calls the *Zero-One Principle*, allows us confirm that a comparator network will sort any of the $n!$ permutations of n letters by simply considering the 2^n n-bit binary words.

Proposition 5.1 (The Zero-One Principle). *If a network will route any "binary" pebbling $\beta : V(G) \to \{0, 1\}$ (with the standard order relation, namely $0 \leq_\beta 1$ and $1 \not\leq_\beta 0$) such that after the routing we have $r \circ \beta(v) \leq_\beta r \circ \beta(w)$ whenever $v <_V w$ for any vertices $v, w \in V(G)$, then it will route any pebbling ϕ so that $r \circ \phi(v) \leq_\phi r \circ \phi(w)$ whenever $v <_V w$.*

Proof. It is easier to prove the contrapositive of this statement, which states that if a network fails to route some pebbling properly then there must be a binary pebbling that is also not routed properly.

Let us denote the vertex set $V(G)$ by $\{v_1, v_2, \ldots v_n\}$, so that the indices indicate the order relation. (So $v_i <_V v_{i+1}$ for all $1 \leq i < n$.) Then we wish to route any pebbling $\phi : V \to P$ such that $r \circ \phi(v_i) \leq_\phi r \circ \phi(v_j)$ for all $i \leq j$. Suppose for a given pebbling ϕ, the network generates a routing r such that $r \circ \phi(v_{k+1}) \leq_\phi r \circ \phi(v_k)$ for some $1 \leq k < n$. We will construct a binary pebbling that is also not routed properly.

Let f be a function on P whose co-domain has some order relation $\leq_f$ such that $f(p) \leq_f f(q)$ whenever $p \leq_\phi q$ for any $p, q \in P$. (The function f is monotonic under $\leq_f$ with respect to $\leq_\phi$.) Then a comparator network that transforms the pebbling ϕ into $r \circ \phi$ will transform $f \circ \phi$ into $f \circ r \circ \phi$, since by construction the same sequence of exchanges is made by the network in both cases.

So, define a function $\beta : V \to \{0, 1\}$ as

$$\beta(w) = \begin{cases} 0 & \text{whenever } \phi(w) \leq_\phi \phi(v_k) \\ 1 & \text{whenever } \phi(w) \not\leq_\phi \phi(v_k) \end{cases} .$$

Since β is a monotonic function with respect to ϕ, the pebbling β must be routed such that $r \circ \beta(v_{k+1}) \leq_\beta r \circ \beta(v_k)$, so β is a binary pebbling that is not sorted by the network. ∎

5.1.2. *When Comparator Networks Might Be Useful (and when they are not).* Given the apparent utility of the Zero-One Principle, it would seem that we have greatly reduced the difficulty of determining the routing number of a given graph. Unfortunately, we find that comparator networks will not suffice for a wide variety of graphs.

Proposition 5.2. *For any graph G, if* $\mathrm{rt}(G) < \log_2 |V(G)|$ *then there exists no comparator network that will route all permutations on G in* $\mathrm{rt}(G)$ *steps.*

Proof. Note that if a comparator network routes all permutations on G, then for any two vertices v, $w \in V(G)$ there exists a path $\langle v, e_1, v_1, e_2, \ldots, w\rangle$ in G such that $e_i \in M_i$,

FIGURE 5. A graph that cannot be optimally routed by comparator networks.

since the network must be able to handle the permutation that takes v to w. Moreover, on each stage in the network, each vertex is incident with at most one edge in the potential matching, so if there is a path from v to k vertices after the i^{th} step, there is a path from v to at most $2k$ vertices after the $i+1^{th}$ step. (That is, each of the k vertices can be matched with at most one vertex.) Hence for an j stage network, there is a path with $e_i \in M_i$ from each vertex to at most 2^j other vertices, so a comparator network must have at least $\log_2 |V(G)|$ stages to route all permutations on G. ∎

Unfortunately, simply having $\operatorname{rt}(G) \geq \log_2 |V(G)|$ does not necessarily allow us to construct comparator networks that will route all permutations of $V(G)$ in $\operatorname{rt}(G)$ matchings.

Example 5.1. Consider the graph H defined by

$$V(H) = \{a, b, c, d\} \text{and } E(H) = \{\{a, b\}, \{a, c\}, \{a, d\}\}$$

(see Figure 5). By inspection, we can show that $\operatorname{rt}(H) = 4$, and clearly $\log_2 |V(H)| = 2 < 4$, but no 4 stage comparator network will route all permutations on H.

Suppose we have a 4 stage comparator network for routing on H. Every matching on H must consist of a single edge, since every edge in H has a as an endpoint. So, any 4 stage network that has an edge in some matching incident with every vertex can include at most one edge more than once. Hence there must be two edges that are included exactly once in the sequence of potential matchings. Let us assume, without loss of generality, that $\{a, b\}$ and $\{a, c\}$ are these edges and that $\{a, b\}$ appears before $\{a, c\}$ in the sequence. Then any permutation that maps c to b will not be properly routed by the network.

Despite these limitations, there are some graphs for which comparator networks will suffice, as we see in the following sections.

5.1.3. *The Odd-Even Transposition Sort for Routing on Paths.* There is a class of graphs, the path on n vertices, P_n, for which comparator networks are capable of achieving all permutations in at most the routing number of steps. (These networks will not yield truly *optimal* routings for all permutations—for some a faster routing is possible—but every permutation is properly routed after $\mathrm{rt}\,(P_n)$ steps.)

FIGURE 6. The paths P_3, P_4 and P_5.

Definition 5.1. The *path on n vertices*, denoted P_n, is the graph with the form

$$V\,(P_n) = \{v_1, v_2, \ldots v_n\} \text{ and } E\,(P_n) = \{\{v_i, v_{i+1}\} : 1 \leq i < n\}\,.$$

Some examples of P_n are shown in Figure 6. Knuth demonstrates that every permutation of $V(P_n)$ can be routed in at most n steps using a comparator network composed of odd edges in odd matchings and even edges in even matchings. That is, each potential matching is defined as

$$M_i = \{\{v_j, v_{j+1}\} \in E\,(P_n) : j \text{ has the same parity as } i\} \text{ for all } 1 \leq i \leq n.$$

FIGURE 7. An odd-even transposition sort on P_3 and P_4. (Dashed, arrowed lines represent edges that were considered, but not used in a matching.)

This algorithm is fairly well known within computer science as the *odd-even transposition sort.* (Note that an *even-odd transposition sort*, defined by

$$M_i = \{\{v_j, v_{j+1}\} \in E(P_n) : j \text{ has the opposite parity of } i\} \text{ for all } 1 \leq i \leq n,$$

will also achieve any permutation on P_n in at most n steps). Moreover, Knuth shows that this network is optimal for the permutation defined by $\pi(v_i) = v_{n+1-i}$ (the permutation that "inverts" the order relation of the vertices), so that $\mathrm{rt}(P_n) = n$ for $n > 2$. Two examples of this network in action are shown in Figure 7.

We will use Knuth's results to analyze a similar class of graphs.

5.2. **Routings on Cycles.** A common pattern for small scale computer networks resembles geometric polygons, with the computers (logically, if not physically) arranged in a circle with communications links joining adjacent computers. Since there are two distinct paths from one computer to every other, this structure is often used for local area networks (LANs) that might, for example, connect several computers to a single printer. Then if a single computer or communication link breaks down, every computer still has access to the printer and other computers. It is somewhat natural then to consider our routing problem on this family of graphs.

FIGURE 8. The cycles C_3, C_4 and C_5.

Definition 5.2. For each $n > 2$, there is a unique (up to isomorphism) connected graph with n vertices that is regular of degree 2. We shall call this graph *the cycle on* n *vertices* and denote it C_n. The cycle on n vertices has the form

$$V(C_n) = \{v_0, v_1, \ldots v_{n-1}\} \text{ and } E(C_n) = \left\{\{v_i, v_{i+1(\bmod n)}\} : 0 \le i \le n\right\}.$$

Note that C_1 and C_2 are undefined, since no simple graph on fewer than three vertices is regular of degree 2.

Some examples of C_n for small n are shown in Figure 8.

For even n, a simple extension of the odd-even transposition sort algorithm will route on C_n in a manner that achieves the routing number.

Theorem 5.3. $\mathrm{rt}(C_n) \le n - 1$ *for even* n*.*

Proof. Let us use the indices of the vertex set to as the order relation $<_V$, so that $v_i <_V v_j$ whenever $i < j$. Consider the $n-1$ stage comparator network whose potential matchings are defined by

$$M_i = \begin{cases} \{\{v_{n-1}, v_0\}, \{v_1, v_2\}, \ldots, \{v_{n-3}, v_{n-2}\}\} & \text{for } i = 1 \\ \{\{v_0, v_1\}, \{v_2, v_3\}, \ldots, \{v_{n-2}, v_{n-1}\}\} & \text{for } i = 2, 4, \ldots, n-2 \\ \{\{v_1, v_2\}, \{v_3, v_4\} \ldots, \{v_{n-3}, v_{n-2}\}\} & \text{for } i = 3, 5, \ldots, n-1 \end{cases}.$$

By the Zero-One Principle, it suffices to show that this algorithm will route any pebbling $\beta : V \to \{0,1\}$, such that $r \circ \beta(v_i) = 0$ if and only if $r \circ \beta(v_j) = 0$ for all $j < i$ and $r \circ \beta(v_i) = 1$ if and only if $r \circ \beta(v_k) = 1$ for all $k > i$.

Consider the action of the network on the "binary pebbles" on vertices v_0 and v_{n-1}. We will consider three cases:

Case 1: If $\beta(v_0) = 0$, we may ignore v_0, since the edges $\{v_0, v_1\}$ and $\{v_0, v_{n-1}\}$ will never be included in a matching. The remaining edges then define an odd-even sort on the path induced by the $n-1$ remaining vertices and hence by Knuth's result can be completed in at most $n-1$ steps.

Case 2: If $\beta(v_{n-1}) = 1$, we may ignore v_{n-1}, since the edges $\{v_{n-2}, v_{n-1}\}$ and $\{v_0, v_{n-1}\}$ will never be included in a matching. The remaining edges then define an even-odd sort on the path induced by the $n-1$ remaining vertices and hence can be completed in at most $n-1$ steps.

Case 3: If $\beta(v_0) = 1$ and $\beta(v_{n-1}) = 0$, then by construction we have $\{v_{n-1}, v_0\} \in m_1$, so $m_1 \circ \beta(v_0) = 0$ and $m_1 \circ \beta(v_{n-1}) = 1$. Thereafter, we may ignore both v_0 and v_{n-1} since the edges adjacent to these vertices will never be used again. The next $n-2$ matchings define an odd-even sort on the remaining path on $n-2$ vertices, so all n vertices are sorted in a total of $n-1$ steps. ∎

To show that the upper bound given in Theorem 5.3 is actually the routing number, we will the need the following definition.

Definition 5.3. For a given permutation π, the vertex $v \in V(G)$ is said to be *permanently fixed* after j steps of the routing r if $r_k \circ \pi(v) = v$ for all $k \geq j$.

Clearly a permutation has not been achieved by a routing until all vertices are permanently fixed. Also, once a vertex is permanently fixed, no edge incident with it is included in any later matching.

Let us now give a name to the 'directions' around a cycle.

Notation 5.1. The pebble on the vertex $v_i \in C_n$ is said to move *clockwise* in a matching m if $\left\{v_i, v_{i+1(\mathrm{mod}\, n)}\right\} \in m$ and is said to move *counter-clockwise* in m if $\left\{v_i, v_{i-1(\mathrm{mod}\, n)}\right\} \in m$.

With these notions in hand we are ready to determine a lower bound for $\mathrm{rt}\,(C_n)$.

Theorem 5.4. $\mathrm{rt}(C_n) \geq n-1$ *for all* n.

Proof. Consider the permutation π defined by $\pi(v_i) = v_{i+1 \bmod n}$. (This permutation just "rotates" the pebbles one vertex clockwise.)

The routing $\langle \{\{v_{n-1}, v_{n-2}\}\}, \{\{v_{n-2}, v_{n-3}\}\}, \ldots \{\{(v_1, v_0)\}\}\rangle$ will achieve π on C_n in $n-1$ steps by permanently fixing one pebble in each of the first $n-2$ matchings, and two pebbles on the last. We will show that no routing is any faster.

Note that initially no vertex is fixed and that each pebble must travel at least $n-1$ steps counter-clockwise (or 1 step clockwise) to reach its destination.

Let us assume (without loss of generality) that the first vertex to be permanently fixed is v_0, and that v_0 is first fixed by the k^{th} matching in the routing (for some $1 \leq k < n-1$). Let p denote the pebble whose destination is v_0. Then p must have moved clockwise in the k^{th} matching and so $\{v_{n-1}, v_0\}$ must have been the edge that moved p. Let's use q to denote the pebble that was on v_0 after $k-1$ matchings. We know that q moved counter-clockwise on the k^{th} step, and must continue to move counter-clockwise until it reaches its destination (since v_0 is permanently fixed). But q must be at least $n-1-k$ counter-clockwise steps from its destination, since it was $n-1$ counter-clockwise steps away initially, and moved at most one edge in each of the k matchings we have already applied. Hence we need at least $n-1-k$ more matchings to fix q and so we need at least $n-1$ steps all together. ∎

We have now completely determined the routing number for even cycles.

Corollary 5.5. $\mathrm{rt}(C_n) = n-1$ *for even* n.

Also, since any routing that can be performed on P_n can be performed on C_n, we have pinned the routing number for odd cycles down to one of two values.

Corollary 5.6. *$n-1 \leq \mathrm{rt}(C_n) \leq n$ for odd n.*

If the "extra edge" in the cycle is actually doing anything, it seems reasonable to make the following conjecture.

Conjecture 5.7. *$\mathrm{rt}(C_n) = n-1$ for all n.*

Although we can see by inspection that it is true for $n = 3, 5, 7$ and 9 (see Appendix A.2), this claim is difficult to prove. Since we must omit at least one vertex from every matching in an odd cycle, it seems unlikely that any comparator network will suffice. Yet if Conjecture 5.7 were false, it would imply that there is some permutation that is a "worst-case" on each of the n different paths we can obtain from C_n by deleting an edge, which doesn't seem to be true, and that cutting the diameter of a path (nearly) in half has no bearing on the routing number for paths with an odd number of vertices.

Moreover, for $n = 3, 5, 6, 7, 8$ and 9, there are exactly two permutations that require $n-1$ matchings to achieve. (These permutations are the 'rotations' by one vertex, i.e. $\pi_1(v_i) = v_{i+1 (\bmod n)}$ and $\pi_2(v_i) = v_{i-1 (\bmod n)}$.) It seems unlikely that this remarkable pattern would be broken for some $n \geq 10$.

6. Routings on Rectangular Meshes

A common design for parallel computer architectures is to lay the processors out on a grid, with communication links between a processor and its immediate neighbors. This structure allows for a large number of interconnections between processors without forcing the communications lines to cross when laid flat. This structure, and multi-dimensional generalizations of it, is used for Cray Research Corporation's Cray-2, Thinking Machines' Connection Machine, Intel's iPSC/860, and many other commercially available parallel computers.

Figure 9. The rectangular meshes $M_{2,3}$, $M_{3,4}$ and $M_{4,4}$.

Definition 6.1. The *m by n rectangular mesh*, denoted $M_{m,n}$ is the graph isomorphic to $P_m \times P_n$.

In order to facilitate the discussion, let us construct some notational conventions for $M_{m,n}$. Since $M_{m,n} \simeq P_m \times P_n$, the mesh can be viewed as the union of n copies of P_m (or as m copies of P_n), with additional edges inserted between the corresponding vertices of each copy. Let us label the vertices within each path P_k with the integers $1, 2, \ldots, k$ so that there is an edge $(i, i+1)$ for each $1 \leq i < k$. Then vertices in the product graph $M_{m,n}$ are labeled by ordered pairs of integers, which we can characterize as

$$V\left(M_{m,n}\right) = \left\{i \times j : 1 \leq i \leq m \text{ and } 1 \leq j \leq n\right\}.$$

It is possible to partition the edge set of $M_{m,n}$ into two sets

$$E_M = \{\{i \times j, (i+1) \times j\} : 1 \leq i < m \text{ and } 1 \leq j \leq n\}$$

and

$$E_N = \{\{i \times j, i \times (j+1)\} : 1 \leq i \leq m \text{ and } 1 \leq j < n\}$$

so that each edge in E_M joins two vertices in same copy of P_m and each edge in E_N joins two vertices in the same copy of P_n. Then

$$E(M_{m,n}) = E_M \cup E_N.$$

Some examples of this labelling are illustrated in Figure 9.

It is not difficult to construct the following lower bound for routings on meshes.

Theorem 6.1. *$\mathrm{rt}(M_{m,n}) \geq m + n - 1$ for $m, n \geq 2$.*

Proof. Consider a permutation π of $V(M_{m,n})$ that maps the "upper-right corner" of the mesh to the "lower-left corner", so that $\pi(m \times n) = 1 \times 1$, $\pi((m-1) \times n) = 1 \times 2$, and $\pi(m \times (n-1)) = 2 \times 1$, as illustrated in Figure 10.

Suppose we can achieve this permutation in a $m + n - 2$ time routing r. Since $\mathrm{d}(m \times n, 1 \times 1) = m + n - 2$, we know that $p_{1\times 1}$ must move closer to 1×1 in each matching and that we must match 1×1 with either 1×2 or 2×1 on the last step.

Without loss of generality, let us assume that after $m + n - 3$ steps, we must have $p_{1\times 2}$ on 1×1, and $p_{1\times 1}$ on 1×2. Then in each of the first $m + n - 3$ matchings, $p_{1\times 2}$ moved closer to 1×1 and $p_{1\times 1}$ moved closer to 1×2.

Note that:

- Both $p_{1\times 1}$ and $p_{1\times 2}$ must be routed along minimally short paths, and so neither pebble can move "right" or "up", when the mesh is oriented as in Figure 10. (That

FIGURE 10. Part of a "bad" permutation on $M_{m,n}$.

is, if $\pi_i^{-1}(1 \times 1) = a \times b$ and $\pi_{i+1}^{-1}(1 \times 1) = c \times d$, then $c \leq a$ and $b \leq d$ for all $0 \leq i < m + n - 3$, and similarly for $\pi_i^{-1}(1 \times 2)$.)

- For every vertex $a \times b \in V(M_{m,n})$ with $a \leq m - 1$, we have $\mathrm{d}(m \times n, a \times b) = \mathrm{d}((m-1) \times n, a \times b) + 1$.
- Every path in the mesh from $(m-1) \times n$ to 1×1 must share some vertex with every path from $m \times n$ to 1×2, since either 1×2 is in the first path or $m \times n$ is on one side and 1×2 is on the other.
- Every "sub-path" of a minimally short path must also be minimally short. (So that if $\langle v_0, e_1, v_2, \ldots e_k, v_k \rangle$ is a minimally short path from v_0 to v_k then $\langle v_0, e_1, v_2, \ldots e_j, v_j \rangle$ is also minimally short for all $0 \leq j \leq k$.)

We will show that no matter which paths we choose, any $m + n - 3$ step routing must match both $p_{1\times 1}$ and $p_{1\times 2}$ with the same vertex on some matching, a contradiction.

Let v denote a vertex that is in both paths. Then $p_{1\times 2}$ must have arrived at v at the $\mathrm{d}((m-1) \times n, v)^{th}$ step of the routing, since the path from $m - 1 \times n$ to v must be minimally short. Then $p_{1\times 1}$ must arrive at v in the $\mathrm{d}(m \times n, v)^{th}$ step of the routing. But since $\mathrm{d}(m \times n, v) = \mathrm{d}((m-1) \times n, v) + 1$, the pebble $p_{1\times 2}$ must leave v in the same matching that $p_{1\times 1}$ arrives at v, and we cannot exchange the two pebbles since that would move one farther away from its destination.

Hence no routing can move all the pebbles in this permutation to their destinations in fewer than $m+n-1$ steps. ∎

Note that since $M_{1,n} \simeq P_n$, the theorem also holds for $m = 1$, $n > 2$.

Since $M_{2,n} \simeq P_2 \times P_n$, we have now bounded $\mathrm{rt}(M_{2,n})$ to one of two values, by applying the Remarkable Cartesian Product Bound (Theorem 4.5).

Corollary 6.2. $n+1 \leq \mathrm{rt}\,(M_{2,n}) \leq n+2$.

Moreover, since $\mathrm{diam}\,(M_{m,n}) = m+n-2$, we know that these graphs fail to meet the Elusive Diameter Bound (Proposition 4.1).

Corollary 6.3. $\mathrm{rt}\,(M_{m,n}) \geq \mathrm{diam}\,(M_{m,n}) + 1$ *(for $m, n \geq 2$).*

Noting that $\mathrm{rt}(P_k) = k$ (for $k > 2$), it seems reasonable to suggest the following.

Conjecture 6.4. $\mathrm{rt}\,(M_{m,n}) \geq m+n$ *for* $m, n > 2$.

The permutation that maps $a \times b$ to $(m+1-a) \times (n+1-b)$ seems a likely candidate for proving this bound, since it would be a "worst-case" permutation in each of the "copies" of P_m and P_n in the mesh. The anomaly for $m = 2$ or $n = 2$ arises from the fact that $\mathrm{rt}(P_2) = 1$.

7. Towards a Tighter Diameter Bound

Let us return now to the question of characterizing graphs which actually meet the Elusive Diameter Bound, that is, we wish to characterize the graphs G for which $\mathrm{rt}\,(G) = \mathrm{diam}\,(G)$.

We have a rather short list of graphs which achieve the bound, the graph with exactly one vertex and the graph with exactly two vertices. In my year of study, these are the only two graphs I have come across that meet the condition.

In contrast, it's easy to find graphs that fail to meet the bound. To begin with, we have the complete graphs K_n and the paths P_n (both for $n > 2$). For every connected graph G with $|V(G)| = n$, it is easy to see that $\mathrm{diam}(K_n) \leq \mathrm{diam}(G) \leq \mathrm{diam}(P_n)$, so K_n and P_n minimize and maximize, respectively, the diameter for graphs on n vertices. (They also maximize and minimize the vertex degree for connected graphs, so this may be somewhat misleading.) In [4], it is shown that a graph with n vertices that contains no cycles (a *tree*, T_n) requires at least n steps, and clearly the diameter of such a graph is at most $n-1$.

By Corollary 4.4, we know that if $\mathrm{diam}\,(G) = 2$ then $\mathrm{rt}\,(G) \geq 3$, so we can eliminate that class of graphs as well. Theorem 5.4 demonstrates that $\mathrm{rt}\,(C_n)$ can be made arbitrarily far from $\mathrm{diam}\,(C_n)$ for large enough n. In Corollary 6.3, we see that $\mathrm{rt}\,(M_{m,n}) > \mathrm{diam}\,(M_{m,n})$, for all $n, m \geq 2$. It doesn't take long for one to suspect that no graph on more than two vertices meets the Elusive Diameter Bound.

Conjecture 7.1. $\mathrm{rt}\,(G) > \mathrm{diam}\,(G)$ *for all graphs G with $|V(G)| > 2$.*

To prove this conjecture, it would be useful to have a framework that can characterize all graphs with given diameter. One such framework is known in the literature ([6]) as the *graphical join.*

Definition 7.1. Given a graph G with the vertex set $\{v_1, v_2, \dots v_s\}$ and a sequence of graphs $H = \{H_1, H_2, \dots, H_s\}$, where $|V(G)| = |H|$, *the graphical join of G relative to H*, denoted $\mathcal{J}(G, H)$ is the graph formed by the union of the graphs in H, with additional

FIGURE 11. A graphical join.

edges between vertices in H_i and vertices in H_j if $(u_i, u_j) \in E(G)$. That is,

$$V(\mathcal{J}(G, H)) = \bigcup_{1 \le i \le s} V(H_i)$$

and

$$E(\mathcal{J}(G, H)) = \{ \bigcup_{1 \le i \le s} E(H_i)\} \cup \{(v, w) : v \in H_i \ , \ w \in H_j \ and \ (v_i, v_j) \in E(G)\} .$$

Note that the *graphical wreath product* is the special case of Definition 7.1 for which every graph in H is the same. A small example of the graphical join is illustrated in Figure 11.

Now, every graph with diameter k can be seen as a vertex-covering subgraph of a generalized join in the form $\mathcal{J}(P_{k+1}, \{H_1, H_2, H_3, \ldots, H_{k+1}\})$ where each H_i is a complete graph with at least one vertex.

To construct the graphical join in this form that corresponds to a given graph G, we let v be some vertex in $V(G)$ such that $\mathrm{d}(v, w) = k$ for some $w \in V(G)$. (Such a vertex exists, by the definition of the diameter.) Then $H_i = \{x \in V(G) : \mathrm{d}(v, x) = i - 1\}$ for all

$1 \le i \le k+1$. In fact, this construction allows us to make an additional restriction upon H_1, since v is the only vertex that is 0 edges away from v. Hence, $H_1 = \{v\}$.

If we can show that $\mathrm{rt}\,(\mathcal{J}\,(P_{k+1}, H)) > k$ for any sequence of graphs H, we have proven Conjecture 7.1. Although I have yet to come up with a full proof, the following weaker theorem can be shown in a manner similar to that used in Section 6.

Theorem 7.2. *Let $k > 2$ and let $G = \mathcal{J}\,(P_{k+1}, \{H_1, H_2, H_3, \ldots, H_{k+1}\})$ be a graphical join for which each H_i is a graph with at least one vertex. If $|V(H_{k+1})| \ge |V(H_1)| + |V(H_2)|$ then $\mathrm{rt}(G) > k$.*

Proof. Since $|V(H_{k+1})| \ge |V(H_1)| + |V(H_2)|$, there exists a permutation π such that $\pi^{-1}(V(H_1) \cup V(H_2)) \subset V(H_{k+1})$. (Every pebble whose destination is in $V(H_1) \cup V(H_2)$ starts in $V(H_{k+1})$.) This permutation cannot be achieved in fewer than $k+1$ steps.

Suppose some routing achieves π in k matchings. Then the pebbles whose destination is in $V(H_1)$ must move closer to their destination in each of the k matchings, since each of these begin in $V(H_{k+1})$.

Since all pebbles are to be fixed after k matchings, we must exchange the pebbles that move into $V(H_1)$ in the k^{th} matching with pebbles whose destination is in $V(H_2)$. So after $k-1$ matchings, we need every pebble in $V(H_1)$ to have a destination in $V(H_2)$. But each of these pebbles started in $V(H_{k+1})$ and so each was originally at least k steps away from $V(H_1)$. Hence no such routing is possible. ∎

Note that there are many graphs that do not meet the conditions of Theorem 7.2 for which Conjecture 7.1 still holds true. For $1 \le k \le 2$, we may apply Proposition 4.3. Also, the paths, cycles and meshes are graphs for which the 'graphical join representation' fails to meet the given condition. Appendix A.2 lists some examples of graphical joins involving complete graphs.

8. Concluding Remarks

Rather than reviewing the theorems we've already seen, I thought that I would take this opportunity to discuss some results that seem to be lurking just at the horizon and some ideas that didn't quite pan out.

For the latter, it seems that the group theoretic structure of the permutations we have been considering has not really been exploited here or in any of the research I've seen. It is intriguing, for example, that the "worst-case" permutations for the cycles all seem to be elements of the dihedral group D_n (the group of symmetries of a regular n-gon, e.g. reflections and rotations), when we label the vertex set in the "reasonable way", as we did in Section 5.2. Part of the difficulty in this algebraic approach is determining precisely what a "reasonable labelling" of the vertex set is, especially for graphs that lack the symmetric structure of C_n. Similarly, the structure of the Cayley digraphs discussed in Section 3.4 seems ripe for exploration.

For the former, due to their prevalence in the parallel computing market, and in 'routing' problems in general, it would be very nice to have a greater understanding of routings on the hyper-cubes, Q^n, the family of graphs isomorphic to Cartesian product of P_2 with itself n times. (For example, $Q^3 = P_2 \times P_2 \times P_2$.) In fact, I believe it may be possible to show that $\mathrm{rt}(Q^n) = n + 1$ for $n \geq 2$, although I'm not quite confident enough to actually state it as a conjecture.

For the Cartesian product of just two paths, i.e. the meshes $M_{m,n}$, I am willing to state the following, though it seems rather difficult to justify.

Conjecture 8.1. $\mathrm{rt}\,(P_m \times P_n) = \mathrm{rt}\,(P_m) + \mathrm{rt}(P_n)$ *for* $m \geq 2$ *and* $n > 2$.

The difficulty here is with permutations that map vertices from one copy of P_m into the same copy of P_n, that is, one must show that it is possible to "untangle" the permutations so that we effectively need only route once in each coordinate.

Finally, considering the apparent difficulty in computing the routing number for specific classes of graphs, and the "combinatorial explosion" of the number of cases to be considered as the size of the vertex and edge sets grow, it would be interesting to explore the NP status of determining the routing number for an arbitrary graph.

References

[1] A. Alon, F.R.K. Chung and R.L. Graham, Routing Permutations on Graphs via Matchings (extended abstract), in the *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, San Diego, California, May 1993, p. 583-591.

[2] A. Alon, F.R.K. Chung and R.L. Graham, Routing Permutations on Graphs via Matchings, *SIAM Journal of Discrete Mathematics*, Vol. 7, No. 3, August 1994, p. 513-530.

[3] D. Knuth, *Art of Computer Programming, Vol. 3: Sorting and Searching*, Addison-Wesley, 1973.

[4] Roberts, A., Symvonis, A., Zhang, L., Routing on Trees via Matchings, Technical Report 494, Basser Department of Computer Science, University of Sydney, January 1995. (available via ftp at ftp://ftp.cs.su.oz.au/pub/tr/TR95_494.ps.Z or http://www.cs.su.oz.au/Publications/techreports.html).

[5] Høyer, P., Larsen, K., Permutation Routing via Matchings, Odense University, Denmark, June 10, 1996. (available via ftp from http://www.emc.dk/DAN/imada/publications.html)

[6] Balińska, K.T. and Quintas, L.V., Two Problems on Self-Complementary Graphs, *1996 Prague Midsummer Combinatorial Workshop (M. Klazar, ed.)*. KAM Series 97-339, Dept. of Applied Mathematics, Charles University, Prague, 1997, p. 8-14.

Appendix A.

A.1. **A Minimal Routing Discovery Program.** Mathematics is a science in which hypothesis takes the form of conjecture and experiment takes the form of rigorous proof. The foundation of hypothesis, of course, is experience and in mathematics experience is found in the solution of examples.

Toward this end, one of the first tasks I undertook when approaching the permutation/routing problem was to completely solve some small cases. This turned out to be a rather tedious process. On a graph as small as C_4, we have 24 permutations and 6 unique matchings to try at any step. The 120 permutations of 5 letters already become prohibitively large for hand computation.

I soon realized that to gain any real intuition about routing on non-trivial graphs, I must find a way to automatically generate routings for specific cases, and hence by examination determine the routing number for specific graphs. I created a program to perform an exhaustive, brute force search for minimal length routings on a given graph. (The search is exhaustive in the sense that it finds a minimally short routing for every permutation, not that *every* minimally short routing is generated for a given permutation.) The program essentially applies all matchings to a permutation, then applies all matchings to the newly generated permutations, and so on, until all permutations are generated. The number of passes through the loop is then the routing number. This program allowed me to isolate fairly quickly the important (some difficult, some easy) permutations and to notice patterns within classes of graphs.

The program was meant to guide my intuition, not to provide a viable method of solution to the general problem, hence there is a great deal of room for optimization here. The program stores all the permutations in memory and as a result is limited to about 9 vertices. (The actual limit is dependant upon the number of vertices and matchings in the graph, as well as the use of system resources by other programs.) A limit of $362,880$ permutations isn't that shabby, after all, and is certainly much more than I (or anyone) would have been able to determine by hand. The following is a simplified outline for the program (some optimization and other details are left out).

`Routine DetermineMatchings`:

- Initialization:
  - Let E be a list of edges in the graph, input by the user.
  - Let M be a list of matchings in the graph, initially empty.
  - Let R be a queue of recently generated matchings, initially containing only the null-matching (the empty set of edges).
- Primary Loop:
  - While R is not empty:
    - Pop the next element from R into a variable, r.
    - For each $e \in E$:
      - If $r \cup \{e\}$ is vertex disjoint, let $r' = r \cup \{e\}$, and append r' to both R and M.

When `DetermineMatchings` terminates, M contains all non-empty matchings in G, ordered by size (so that $|m_i| \leq |m_{i+1}|$ for all subsequent matchings m_i, $m_{i+1} \in M$).

`Routine DetermineRoutings`:

- Initialization:
  - Let M be a list of matchings in the graph, generated by the routine `DetermineMatchings`.
  - Let L be a list of generated permutations, initially containing only the identity permutation. Each element $l \in L$ consists of the permutation l_π, and the routing l_r that achieved it.
  - Let Q be a list of recently generated permutations, initially containing only the identity permutation.
- Primary Loop:
  - While Q is not empty:
    - Pop the next element from Q into q.
      - For each $m \in M$:
      - Let $q'_\pi = m \circ q_\pi$. Let q'_r be q_r with m appended.
      - If q' is not in L, append q' to both Q and L.

When Q is empty, all permutations have been generated. The program then outputs all the permutations (in both the table and orbit-decomposition form) in L, together with their respective routings.

A.2. **Summary of Results for Some Specific Graphs.** These results were obtained with the help of the program outlined above. Great pains were taken to ensure the accuracy of the program—including complete manual verification of results for some small graphs—and no errors have been found, but of course, there is no guarantee that all of these computations are 100% accurate. Even if the software is completely bug-free, there is some small probability that a hardware error has occurred. (Do we really *know* that the Four - Color Theorem is true?) Actually, most if not all of these graphs were run through the program more that once, even on different machines, so the likelihood of hardware anomalies introducing error seems very tiny indeed.

Cycles:

- $\mathrm{rt}(C_n) = n - 1$ for all $3 \le n \le 9$.
- There are exactly two permutations for which $\mathrm{rt}(C_n, \pi) = n - 1$ for $5 \le n \le 9$, the "rotations by one vertex", where $\pi(v_i) = v_{i+1 \bmod n}$ for all i or $\pi(v_i) = v_{i-1 \bmod n}$ for all i. The same is true for $n = 3$. When $n = 4$, there are two additional "worst-case" permutations, $\begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 3 & 2 & 1 \end{pmatrix}$ and $\begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 1 & 0 & 3 \end{pmatrix}$, "reflections" across one diagonal or the other (but not both).

Cartesian Products:

- $\mathrm{rt}(P_2 \times P_2) = 3$. (Note that $P_2 \times P_2 \simeq C_4$.)
- $\mathrm{rt}(P_2 \times P_2 \times P_2) = 4$.
- $\mathrm{rt}(M_{n,2}) = n + 1$ for $2 \le n \le 4$.
- $\mathrm{rt}(M_{3,3}) = 6$

Graphical Joins:

- $\mathrm{rt}\left(\mathcal{J}\left(P_3, \{K_1, K_3, K_2\}\right)\right) = 3$
- $\mathrm{rt}\left(\mathcal{J}\left(P_4, \{K_1, K_2, K_3, K_1\}\right)\right) = 4$
- $\mathrm{rt}\left(\mathcal{J}\left(P_4, \{K_1, K_3, K_3, K_1\}\right)\right) = 4$

A sample output from this program is attached to this document.