International Journal of Networking and Computing – www.ijnc.org
ISSN 2185-2839 (print) ISSN 2185-2847 (online)
Volume 4, Number 1, pages 70–87, January 2014

An Optimal Time-Power Tradeoff for Sorting on a Mesh-Connected Computer with On-Chip Optics

Patrick Poon
Computer Science and Engineering, University of Michigan
Ann Arbor, MI 48109, USA

Quentin F. Stout
Computer Science and Engineering, University of Michigan
Ann Arbor, MI 48109, USA

Received: July 23, 2013
Revised: October 19, 2013
Accepted: December 2, 2013
Communicated by Susumu Matsumae

Abstract

Energy consumption has become a critical factor constraining the design of massively parallel computers, necessitating the development of new models and energy-efficient algorithms. The primary component of on-chip energy consumption is data movement, and the mesh computer is a natural model of this, explicitly taking distance into account. Unfortunately the dark silicon problem increasingly constrains the number of bits which can be moved simultaneously. For sorting, standard mesh algorithms minimize time and total data movement, and hence constraining the mesh to use only half its processors at any instant must double the time. It is anticipated that on-chip optics will be used to minimize the energy needed to move bits, but they have constraints on their layout. In an abstract model, we show that a pyramidal layout and a new power-aware algorithm allows one to sort with only a square root increase in time as the fraction of processors simultaneously powered decreases. Furthermore, this layout is shown to be optimal in terms of the time-power tradeoff required for sorting. Previous algorithms assumed fully powered systems, hence pyramid sorting was of no interest since when fully powered they are no faster than the base mesh. Our results show asymptotic theoretical limits of computation and energy usage on a model which takes physical constraints and developing interconnection technology into account.

Keywords: Parallel Algorithms, Layout, Sorting, Mesh-Connected, All-Nearest-Neighbors, Minimal Spanning Forest

1 Introduction

Power consumption has become an important design consideration for systems ranging from mobile devices to supercomputers. As the number of processing units has increased, so has energy usage, which is a problem when there is a limit on the available total energy or peak power. In addition, increasing transistor densities has brought about physical constraints of heat dissipation, limiting the fraction of chips operating at full speed. This "dark silicon" problem will only worsen [5, 35].

Further, since processors occupy physical volume, there are processors that are far apart from each other that take non-constant time and energy to communicate between. Algorithms must take advantage of locality to reduce time and energy, a fact typically ignored in algorithms for abstract shared-memory models such as the PRAM. Note that for parallel computers the relevant energy concern is peak power consumption from an external source of power, as opposed to the total energy limits of battery-powered devices.

Our goal is to study fundamental limits in the tradeoff of time vs. peak power for basic problems such as sorting. Towards this end we use a classic abstract model of a scalable parallel architecture that addresses the issues of locality and power consumption. While there are many models of distributed-memory parallel architectures, such as hypercubes, it is the mesh that is most relevant. Numerous mesh models have been analyzed and built ever since von Neumann introduced cellular automata [37]. The mesh is a scalable parallel computer architecture and has also been used as a model of many physical processes where locality strongly affects behavior [32, 38]. Here, both roles are intertwined in the consideration of energy consumption in massively parallel computation. Our primary contribution is an optimal sorting algorithm, and proof of optimality, for a 2-dimensional mesh-connected computer augmented by optical interconnections. The algorithm optimizes the time vs. peak power tradeoff throughout the entire range of peak power from 1 to N, where N is the input size.

A 2-dimensional *mesh-connected computer*, or *mesh*, of size N is a parallel computer consisting of N processors arranged in a square lattice. All operations on values stored in a processor's memory, including transmitting a value to a neighbor, take constant time and energy. Technically it should be constant energy per bit, not word, but we will ignore this extra logarithmic factor. Thus the time and energy required to transmit a word of information from one processor to another is linear in the distance between them. Energy can often be viewed as equivalent to the concept of *work* as it is used in parallel algorithms.

A processor is *active* if it is calculating or communicating and is otherwise inactive and not using energy. More precisely, an inactive processor is in a very low power sleep mode and our algorithms consider the power needed above this level. We give algorithms that minimize time given a *peak power* bound, where peak power is the maximum number of processors active at any one time. The fraction of processors that are active will be denoted by r, so peak power is equal to $r \cdot N$. Total energy usage is the integral of power over time.

Much current research focuses on making processors more power efficient and one approach is to add optical connections or lasers onto processors [4, 9, 11, 13, 15, 21, 22, 36]. This technology is promised to bring about many advancements. For example, the article *Interconnect Opportunities for Gigascale Integration* [17] states:

> Microphotonic interconnects have long-term potential to reduce latency, power dissipation, and crosstalk while increasing bandwidth.

Since light is fast relative to electrical connections and suffers less attenuation optical connections have important advantages: they can link processors far apart and transfer data quickly. In addition, power usage is less as transmitting an optical signal scales to long distances while taking nearly the same energy for shorter distances [21, 27]. We model on-chip photonics capabilities by adding "optical" connections or *optics* to the mesh. We refer to the connections in the standard mesh model as *wires*. Communication over an optical connection will take constant time, though not necessarily the same constant as for communication over a wire. However, in O-notation the differences are irrelevant. It is difficult to build multiple layers of optics on a chip so we allow only one layer of optics, and we prohibit crossings [4].

We do not require that this extra layer actually be optical interconnects, merely that it provides the capability of transmitting information long distances with low power relative to the capabilities of standard wires. Whether this is supplied via optical waveguides, carbon nanotubes, or whatever else emerges, is not relevant to our analyses.

In Section 3, we show a lower bound for sorting given any possible layout of optics. Sorting is a fundamental operation which requires extensive communication, and it is a key step in many algorithms. Given a mesh of size N with N items stored one per processor, without optics, sorting

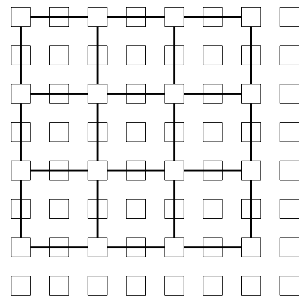


Figure 1: The optical connections of a 4×4 optical mesh on a 8×8 mesh

the items into a specified ordering takes $\Omega(\sqrt{N})$ time and $\Omega(N^{3/2})$ total energy since simple matrix transposition results in items moving a total distance of $\Theta(N^{3/2})$. Furthermore, mesh algorithms achieving these lower bounds have been widely known and refined since the 1970's [6, 12, 16, 19, 20, 24, 25, 26, 31]. By stepwise simulation, it follows that sorting can be accomplished in $\Theta(\frac{1}{r}\sqrt{N})$ time using rN peak power. In Section 5 it will be shown that by adding an optical layer this can be reduced to $\Theta\left(\frac{1}{\sqrt{r}}\sqrt{N} + \frac{1}{r}\log N\right)$, matching the lower bound shown in Section 3. For $r \in \omega\left(\frac{\log^2 N}{N}\right)$ this is a sublinear increase in time as the peak power is decreased. Illustrative applications which exploit sorting and have the same tradeoff are given for two different classes of problems: all-nearest-neighbors in Section 6 and minimum spanning forest in Section 7.

2 The Model

To simplify exposition, let $n = \sqrt{N}$ and assume that n is a power of 2, with modifications to the more general case being straightforward. We start off with the standard mesh model of N processors connected in an $n \times n$ grid, where each processor is connected to its adjacent neighbors. $P(i,j)$ denotes the processor at coordinates (i,j), for $i,j \in \{0,1,\ldots,n-1\}$. Each processor has a fixed number of words of memory, each with $\Theta(\log N)$ bits. This is enough to store its location in the mesh and a constant number of other values. Operations on data stored in a processor's local memory and communication of a word of data to an adjacent processor take constant time and energy. Power is defined to be the number of processors using energy at a given time step. Modeling physical properties, we assume that optics cannot cross and each optical connection has a fixed minimal width. This implies that each processor has at most a constant number of optical connections, the total area used by optics is $O(N)$, and that the bisection bandwidth of any optical network is $O(\sqrt{N})$.

We first consider the simple case of adding only optics of one length to the mesh. Given optics of a specific length, say of length n/k, we can create a $k \times k$ mesh network consisting only of optical connections on top of the wire mesh. This $k \times k$ mesh, which we will call an *optical mesh*, consists of processors $P(i,j)$, for $i,j \in \{0, n/k, 2n/k, \ldots, (k-1)n/k)\}$. Figure 1 illustrates an example. It is useful to note that in each $n/k \times n/k$ submesh there is one processor that is part of the optical mesh and so the communication diameter of the mesh with an optical mesh is $\Theta(\sqrt{N}/k + k)$, which is less than the $\Theta(\sqrt{N})$ communication diameter of a standard mesh for most values of k.

In the more general case where we allow optics of any length, multiple mesh-like optical networks of different sizes can be embedded and a pyramid-like network can be achieved within one layer of optics. We call this an *optical pyramid*, defined as follows: let a be the index of the least significant 1 bit of i. For each row i such that $2 \le a \le \lg n - 1$, connect the processor in column $b \cdot 2^{a-1}$ to the processor in column $(b+1) \cdot 2^{a-1} - 1$, for $0 \le b \le n/2^{a-1} - 1$. Likewise, for each column i such that $2 \le a \le \lg n - 1$, connect the processor in row $b \cdot 2^{a-1}$ to the processor in row $(b+1) \cdot 2^{a-1} - 1$, for $0 \le b \le n/2^{a-1} - 1$. Note that optics are of length $2^{a-1} - 1$ and it is easy to check that none of these optical connections cross. Figure 2 illustrates the layout of the optical connections for a 32×32 mesh. The standard wire connections in the mesh are omitted in the figure and only the

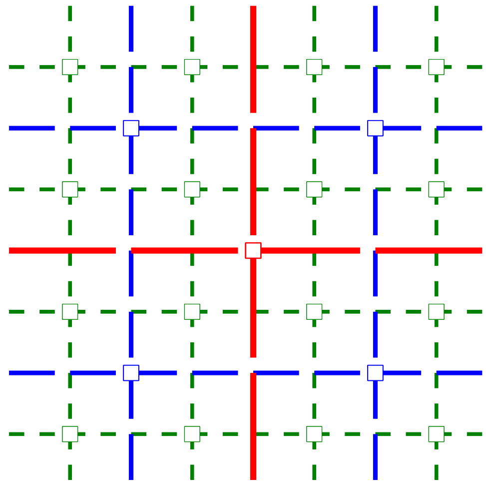
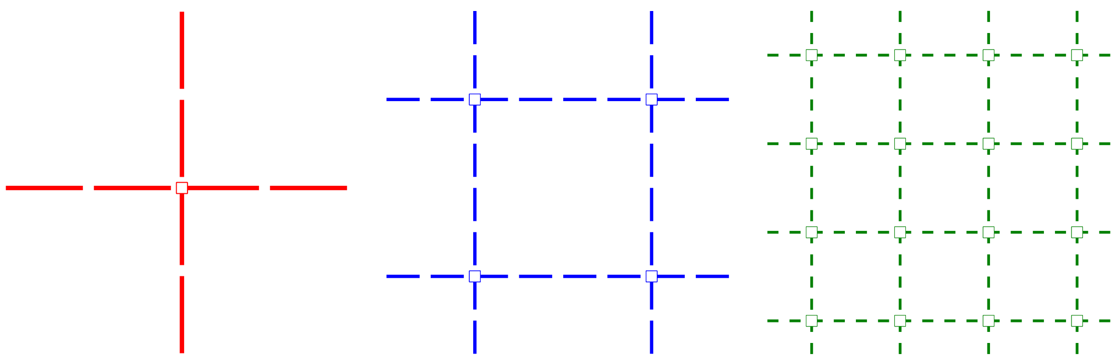


Figure 2: Optical connections and processors in optical meshes of a 32×32 mesh with optical pyramid.



(a) 1×1 optical mesh (optics of length 7)

(b) 2×2 optical mesh (optics of length 3)

(c) 4×4 optical mesh (optics of length 1)

Figure 3: The different levels of the layout of optical connections shown in Figure 2.

processors that compose the optical meshes are shown.

Alternatively, a recursive definition may be easier to understand: on an $n \times n$ mesh, place optical connections of length $n/4 - 1$ so that the following eight pairs of processors are connected:

$$\begin{array}{ll}
P(\frac{n}{2},\, 0),\ P(\frac{n}{2},\, \frac{n}{4}-1) & P(\frac{n}{2},\, \frac{n}{4}),\ P(\frac{n}{2},\, \frac{n}{2}-1) \\
P(\frac{n}{2},\, \frac{n}{2}),\ P(\frac{n}{2},\, \frac{3n}{4}-1) & P(\frac{n}{2},\, \frac{3n}{4}),\ P(\frac{n}{2},\, n-1) \\
P(0,\, \frac{n}{2}),\ P(\frac{n}{4}-1,\, \frac{n}{2}) & P(\frac{n}{4},\, \frac{n}{2}),\ P(\frac{n}{2}-1,\, \frac{n}{2}) \\
P(\frac{n}{2},\, \frac{n}{2}),\ P(\frac{3n}{4}-1,\, \frac{n}{2}) & P(\frac{3n}{4},\, \frac{n}{2}),\ P(n-1,\, \frac{n}{2})
\end{array}$$

Then recursively place eight optics in each of the four $n/2 \times n/2$ submeshes until reaching the base case of a 4×4 mesh after a total of $\lg n - 1$ levels of recursion. Figure 3 illustrates the recursive layout.

While the wire connections are not shown, note that they play an important role in the optical mesh. Since the optical connections are not allowed to cross, the underlying wires are used to transmit information between adjacent endpoints of optical connections. E.g., to pass information from $P(\frac{n}{2},\, 0)$ to $P(\frac{n}{2},\, \frac{n}{2}-1)$ using optics of length $n/4-1$, it is passed from $P(\frac{n}{2},\, 0)$ to $P(\frac{n}{2},\, \frac{n}{4}-1)$ using optics, then to $P(\frac{n}{2},\, \frac{n}{4})$ using a wire, and finally to $P(\frac{n}{2},\, \frac{n}{2}-1)$ using optics.

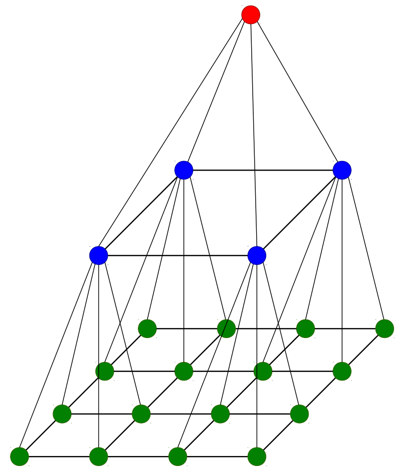


Figure 4: A classical pyramid computer model of size 16.

2.1 Useful Properties

The optical pyramid has a communication diameter of $\Theta(\log N)$, which is far smaller than the $\Theta(\sqrt{N})$ communication diameter of the mesh. Using the tree structure embedded in the pyramid, small amounts of data can be moved across the mesh quickly.

A particularly useful property is the recursive definition of the layout of the optical pyramid: the layout of the optical connections in any square submesh is also a logical optical pyramid. That is to say, an optical pyramid over any square submesh can be simulated with a constant factor overhead. In situations where recursive algorithms are applied to ordered data, the Hilbert space-filling curve ordering is useful as it keeps local data in close proximity. In fact, for items ordered in a Hilbert curve, every set of k consecutive items is contained within a square submesh with a logical optical pyramid of size at most $4k$.

Another property is that optics of the same length compose a network that can simulate an $n/2^{a+1} \times n/2^{a+1}$ optical mesh with a constant factor overhead, for $2 \leq a \leq \lg n - 1$. This is accomplished using processors evenly spaced 2^{a+1} apart in the entire mesh, that is, processors $P(i,j)$, for $i,j \in \{2^a, 2^a + 2^{a+1}, \ldots, 2^a + (n/2^{a+1} - 1)2^{a+1}\}$, are processors in the optical mesh. Specifically, it takes eight time steps to communicate between adjacent processors in an optical mesh; processors need only to send data over four wire connections and four optical connections to reach the next processor. For convenience, we will refer to these as optical meshes, ignoring the constant factor overhead contributed by the gaps between optics. Note that communication from a processor on one optical mesh to a processor on the next smaller or next large optical mesh takes constant time.

Our results utilize the pyramid in new ways. The classical pyramid computer model (structure and communication links shown in Figure 4) is a model that has been considerably studied in the past. However, no previous work on pyramid computer algorithms considered energy usage, that is, there was no penalty for having all processors running all the time. Algorithms for a standard pyramid can be run on the mesh with an optical pyramid using stepwise simulation with a constant factor overhead, though such simple usage in general has a linear tradeoff of time vs. peak power and is of less interest here. Further, for communication-intensive problems such as sorting, the pyramid with all processors active is no faster than the base mesh. Our power aware algorithms achieve a sublinear increase in time as the peak power decreases, a property unachievable if only the base mesh is utilized. However, this requires pyramid algorithms quite different from previous ones.

3 Lower Bounds

In the mesh model, there are two fundamental properties that provide lower bounds. The first is communication diameter, which is relevant when data needs to be moved from one side of the mesh to the other. On a standard mesh of size N, the diameter is $\Theta(\sqrt{N})$. If a problem requires global communication, that is, at least one processor may receive information originated by any processor, then any algorithm that solves the problem takes $\Omega(\sqrt{N})$ time. The other property is the bisection bandwidth, relevant when the problem requires moving all of the data in one half of the mesh to the other half. For the mesh, the bisection bandwidth is $\Theta(\sqrt{N})$, and any algorithm which moves all the data from one half to the other takes $\Omega(\sqrt{N})$ time.

The benefit of optical connections on meshes is the ability to decrease the diameter, effectively lowering the communication lower bound and reducing the total energy required to move data across the mesh. Unfortunately, they do not change the bisection bandwidth lower bound. Thus they can reduce energy usage but cannot reduce running time of problems that require significant bandwidth. As will be shown, adding optical connections allows sorting and permutation to be solved in $o(N^{3/2})$ total energy, surpassing the lower bound for permutation in a standard mesh. The algorithms in this paper run at peak power most of the time, so their total energy is $\Theta(\text{time} \times \text{peak power})$.

In addition to the bisection bandwidth, there are some important lower bounds on meshes with optical connections. These include:

Diameter: no matter how optics are added, the diameter of the mesh is $\Omega(\log N)$. Since each processor is adjacent to only a constant number of processors, the number of reachable processors is at most exponential in the number of time steps.

Permutation Energy: no matter how optics are added, the total energy to permute is $\Omega(N \log N)$, where permutation is defined to be the operation that communicates data from each processor in the mesh to a unique other processor in the mesh. To see this, note that the comments above about the number of reachable processors shows that if processors are connected to at most t others, then in $q = (\log_t N)/2 - 1$ steps, no processor can communicate with more than $\sum_{i=0}^{q} t^i < t^{q+1} = \sqrt{N}$ processors. Given an optical layout, construct the following permutation: go through the processors in row-major order. For each processor, set its destination to be the first processor which is not reachable in r or fewer steps and which is not already the destination of some other processor. It is possible that the final $\sqrt{N}$ processors have no such destination, in which case choose the destination arbitrarily from the processors that are not yet destinations. Thus at least $\Theta(N)$ processors are sending to a destination requiring at least $\Theta(\log N)$ steps. Note this also shows that most permutations require $\Omega(N \log N)$ total energy.

Sorting Energy: there is the obvious $\Omega(N \log N)$ lower bound on energy due to the number of comparisons required. This is also a lower bound on the energy needed for data movement. This follows from the permutation bound since a permutation can always be achieved by sorting, using the destination as the key. Using rN peak power, linear speedup gives an $\Omega(\frac{1}{r} \log N)$ running time lower bound for permutation and sorting.

Note that while all parallel computers have the $\Omega(N \log N)$ lower bound for sorting comparisons, they do not all share this lower bound for permutation. In standard models of shared-memory machines, permutation takes only $\Theta(N)$ operations, while the above proof shows that for distributed-memory machines with bounded degree, the lower bound on data movement is $\Theta(N \log N)$.

3.1 Sorting

We now show a lower bound running time for sorting on any layout of noncrossing optics. In Section 5, a sorting algorithm on the mesh with an optical pyramid is given with a matching upper bound, so the optical pyramid layout is an optimal layout in terms of sorting.

We first need to make a claim about the crossing number of a complete bipartite graph, which we will use in the proof of the lower bound. Given a graph G, the crossing number $\mathrm{cr}(G)$ is defined to be the minimum number of crossing pairs of edges, over all drawings of G in the plane. The following proofs regarding crossing number follow ideas found in [10], but Leighton [14] claims these techniques are well-known. The complete bipartite graph with a bipartition into two sets of vertices,

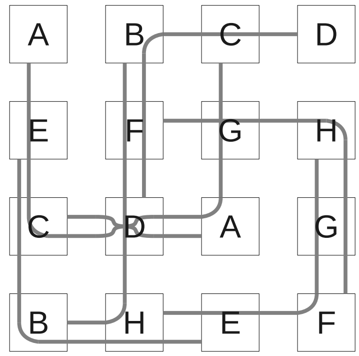


Figure 5: A drawing of an execution of a permutation where pairs of processors with the same label exchange data.

one of size M and the other of size N, is denoted $K_{M,N}$.

Lemma 3.1. *The crossing number of $K_{3,N}$ is $\Omega(N^2)$.*

Proof. We can assume that no two edges that share a common endpoint cross because such a crossing can be removed and still produce a valid drawing.

Given a drawing of $K_{3,N}$, consider the N subdrawings where one of the vertices in the part with N vertices is removed along with its incident edges. Each crossing in the drawing of $K_{3,N}$ exists in $N-2$ of these subdrawings, that is, all of the subdrawings except the two that removed one of the edges in the crossing. The number of crossings in each subdrawing must be at least the crossing number of $K_{3,N-1}$. Therefore, $(N-2)\,\mathrm{cr}(K_{3,N}) \geq N\,\mathrm{cr}(K_{3,N-1})$. We also know that $\mathrm{cr}(K_{3,3}) = 1$ since $K_{3,3}$ is not planar and can be drawn with one crossing. This implies $\mathrm{cr}(K_{3,N}) \geq \frac{N}{N-2} \cdot \frac{N-1}{N-3} \cdot \frac{N-2}{N-4} \cdots \frac{6}{4} \cdot \frac{5}{3} \cdot \frac{4}{2} = \frac{N(N-1)}{6} \in \Omega(N^2)$ □

Lemma 3.2. *The crossing number of $K_{M,N}$ is $\Omega(M^2N^2)$.*

Proof. The argument is similar to the proof of the previous lemma. Consider M subdrawings of $K_{M,N}$ where one of the vertices in the part of the graph with M vertices is removed along with its incident edges. Each crossing in the drawing of $K_{M,N}$ exists in $M-2$ of these drawings. The number of crossings in each subdrawing must be at least the crossing number of $K_{M-1,N}$. Therefore, $(M-2)\,\mathrm{cr}(K_{M,N}) \geq M\,\mathrm{cr}(K_{M-1,N})$. Applying the previous lemma, this implies $\mathrm{cr}(K_{M,N}) \in \Omega(\frac{M}{M-2} \cdot \frac{M-1}{M-3} \cdot \frac{M-2}{M-4} \cdots \frac{5}{3} \cdot \frac{4}{2} \cdot N^2) = \Omega(M^2N^2)$. □

Specifically, we will use the fact that $\mathrm{cr}(K_{N/2,N/2}) \in \Omega(N^4)$ in the following theorem.

Theorem 3.3. *Given a mesh of size N with optical connections and rN peak power, a permutation of N items stored one per processor requires $\Omega\left(\frac{1}{\sqrt{r}}\sqrt{N}\right)$ time.*

Proof. A permutation can be represented by an N vertex graph, where each processor is represented by a vertex and an edge exists between two vertices if there is an item with initial and final positions that correspond to those two vertices. We may exclude items that do not move as they do not require any energy to reach their destination.

Every possible execution of a permutation can be represented by a planar drawing of the edges of the graph corresponding to the permutation. To represent an execution by a drawing, vertices are placed as an $\sqrt{N} \times \sqrt{N}$ lattice in the plane and an edge is drawn through a vertex if that vertex is used in the route taken by the item corresponding to that edge. Note that edges can be drawn such that edges share points at discrete locations and they only share a point when their corresponding routes cross. See Figure 5 for an example of a drawing that corresponds to the execution of a permutation on a 4×4 mesh. Whenever two edges share a common point in such a drawing, other than at an endpoint, one of the corresponding routes in the execution must have incurred one unit of energy usage. While the drawing of an edge does not specify what parts of the route are using wires or optics, it cannot be the case that both routes passed over the common processor using optics as optics cannot cross.

In a given execution, it may be the case that the routes of many items cross at a common location. In the drawing, this would correspond to edges of these routes sharing a common point in the plane. We call a point in the plane that is shared by more than one edge an *intersection*.

Let processors be labeled $P_0, \ldots, P_{N-1}$. Consider $N/2$ permutations, where permutation i moves the item located at P_j to $P_{N/2+(j+i \bmod N/2)}$, for $0 \leq i \leq N/2-1$ and $0 \leq j \leq N/2-1$. An execution of one of these $N/2$ permutations corresponds to a drawing of the graph that represents the permutation. The union of these $N/2$ drawings is a drawing of the complete bipartite graph $K_{N/2,N/2}$ and such a drawing has $\Omega(N^4)$ crossings.

To prove the theorem, it is enough to show that computing all $N/2$ permutations takes a total of $\Omega(N^2/\sqrt{S})$ time, for peak power $S = rN$. We first assume that computing these $N/2$ permutations in total can be accomplished in T time. Therefore, every location in the mesh is used at most T times, and thus any point in a drawing may be shared by at most T edges. This implies $O(T^2)$ pairwise crossings of edges may be located at an intersection.

The total energy required for any execution must be at least the sum of the number of times each intersection is traversed. We will show this is at least the product of the number of intersections and the maximum number of times each intersection is used. Intuitively, the minimum energy is achieved when the number of points of intersection is minimized and each intersection is used the maximum possible number of times.

Let there be m intersections in the planar drawing of these $N/2$ permutations. For each intersection i, let x_i be the number of edges which go through the intersection. We know $x_i \leq T$, energy is at least $\sum_{i=1}^{m} x_i$, and the total number of pairwise crossings is $\sum_{i=1}^{m} \frac{x_i(x_i-1)}{2} \in \Omega(N^4)$. Using the method of Lagrange multipliers, the minimum energy occurs when all x_i have the same value. Let this value be x. This implies that $m \in \Omega(N^4/x^2)$. Energy is at least the number of times each intersection is traversed, which is $\Omega((N^4/x^2) \cdot x) = \Omega(N^4/x)$. Time is at least energy divided by peak power, so $T \in \Omega((N^4/x)/S)$. Combining this with the fact that $x \leq T$, T must be at least $\Omega(N^2/\sqrt{S})$. □

Since permutation can be accomplished by sorting using each item's destination as the key, sorting must also take $\Omega\left(\frac{1}{\sqrt{r}}\sqrt{N}\right)$ time. Combining this with the linear speedup lower bound for sorting, we have the following theorem.

Theorem 3.4. *Given a mesh of size N with optical connections and rN peak power, sorting N items requires $\Omega\left(\frac{1}{\sqrt{r}}\sqrt{N} + \frac{1}{r}\log N\right)$ time.* □

4 Basic Algorithms

The following operations on the mesh with an optical pyramid are frequently used operations in algorithms. We denote peak power by S (which is equal to rN, where r is the fraction of total processors that are active, $1/N \leq r \leq 1$). For convenience, let $s = \sqrt{S}$ and assume that s is a power of 2. This simplifies notation as we will frequently refer to the $s \times s$ optical mesh.

Our analyses in the following sections will be given in terms of the available peak power S. However, running times will often be expressed in terms of r, in order to emphasize the relationship between the amount of slowdown and fraction of processors active.

4.1 Routing

On the standard mesh, given two arbitrary processors in which one sends a word of data to the other, routing the data between the processors takes $\Theta(\sqrt{N})$ time, with the lower bound set by the communication diameter. On the pyramid, data can be communicated in $O(\log N)$ time by using the tree structure of the pyramid. If k processors have data that needs to be sent to other processors, the values at those processors are first moved to the $k \times k$ optical mesh in $O(k + \log N)$ time using the tree structure of the pyramid. Then all values can be routed on the optical mesh to their $n/k \times n/k$ submesh destination. Using pipelining, this can all be accomplished in $O(k + \log N)$ time. More generally, S processors can send data to other processors in $O(\sqrt{S} + \log N)$ time using the $s \times s$ optical mesh as long as the bandwidth on the optical mesh between processors and their destinations is $\Omega(\sqrt{S})$.

4.2 Broadcast and Reduction

On the standard mesh, it is possible to broadcast in $\Theta(\frac{1}{r} + \sqrt{N})$ time. The lower bound comes from the communication diameter and evenly dividing the peak power of the optimal algorithm, and it is easy to achieve this bound. On the pyramid, this can be reduced to $\Theta(\frac{1}{r} + \log N)$. To do this, the value to be broadcast is moved to processor $P(n/2, n/2)$ in $O(\log N)$ time. Then, using the tree network embedded in the pyramid, the value is broadcast to each $n/s \times n/s$ submesh in $O(\log N)$ time. In each submesh, one unit of energy per time step is used to broadcast to all processors in the submesh, taking $\Theta(\frac{1}{r})$ time.

The data movement in a reduction operation is the reverse of broadcast, where values are combined using a semigroup operator. This can also be accomplished in $\Theta(\frac{1}{r} + \log N)$ time.

4.3 Scan

Given values $a_0, \cdots, a_{N-1}$ stored one value per processor on the mesh, and given a semigroup operation $\otimes$ that can be computed in constant time, a scan operation results in processor i having the value $a_0 \otimes a_1 \otimes \cdots \otimes a_i$, for $0 \leq i \leq N-1$.

If the values are ordered by a Hilbert or Z-order space filling curve in the mesh with an optical pyramid, a scan can be computed in $\Theta(\frac{1}{r} + \log N)$ time by executing the standard logarithmic time algorithm that recursively computes prefixes using the tree network embedded in the optical pyramid. If the values are in row-major order, the scan can be computed in $\Theta(\frac{1}{r} + \sqrt[4]{N})$ time by first dividing the mesh into $\sqrt[4]{N} \times \sqrt[4]{N}$ submeshes and computing the scan in each submesh, then combining data using the $\sqrt[4]{N} \times \sqrt[4]{N}$ optical mesh.

5 Sorting

In this section, we show that, given rN peak power, permutation and sorting can be accomplished in $\Theta\left(\frac{1}{\sqrt{r}}\sqrt{N}\right)$ time using the optical pyramid, as opposed to the $\Theta\left(\frac{1}{r}\sqrt{N}\right)$ time required on a standard mesh, for $r \in \Omega\left(\frac{\log^2 N}{N}\right)$. Thus the lower bound given in Section 3.1 can be achieved. We first give an algorithm for permutation and then use it within the sorting algorithm. To simplify notation, algorithms are presented in terms of peak power S. Note that, due to the issue of many items clustered together not having enough bandwidth to be spread out in the desired running time, the algorithm is not as simple as moving S items at a time to their destinations.

Lemma 5.1. *On a mesh of size N with an optical pyramid, N items can be permuted in $\Theta\left(\frac{1}{\sqrt{r}}\sqrt{N} + \frac{1}{r}\log N\right)$ time using rN peak power.*

Proof. Our algorithm uses the $s \times s$ optical mesh to move data across the mesh. We conceptually partition the mesh horizontally into $n/s \times n$ submeshes, referred to as *optic rows*, labeled $A_0, \ldots, A_{s-1}$ and also partition the mesh vertically into $n \times n/s$ submeshes, referred to as *optic columns*, labeled $B_0, \ldots, B_{s-1}$. Every A_i contains a single row of the $s \times s$ optical mesh, and every B_i contains a single column, for $0 \leq i \leq s-1$. Each processor on the optical mesh belongs to a unique pair (A_i, B_j) of submeshes.

Algorithm 1 is an outline of the algorithm. In more detail, the following is repeated N/S times:

Algorithm 1 Permutation algorithm.

for all items; S at a time, s per optic column **do**
  $i_{\text{int}} \leftarrow$ index of optic column of origin
  $i_{\text{dest}} \leftarrow$ index of optic row of destination
  for $i, j \leftarrow 0, s$ **parallel do**
    $count(i,j) \leftarrow$ number of items from B_j with $i = i_{\text{dest}}$
    $x(i,j) \leftarrow \sum_{j'=0}^{j-1} count(i,j')$
    $y(i) \leftarrow \sum_{j'=0}^{s-1} count(i,j')$
    $z \leftarrow$ index within items moved to $(A_{i_{\text{dest}}}, B_j)$
    $j_{\text{int}} \leftarrow \left\lfloor \frac{x(i,j)+z}{y(i)} \cdot s \right\rfloor$
  end for
  Move item to $A_{i_{\text{int}}}$
  Move item to $B_{j_{\text{int}}}$
  Move item to $A_{i_{\text{dest}}}$
end for
for $i \leftarrow 0, s$ **parallel do**
  for all items in A_i; s at a time **do**
    Move item to destination
  end for
end for

- Within each B_j, s items that have not yet been moved are chosen. A copy of each of these s items is moved along the vertical optical connections in B_j to the processor on the optical mesh in $A_{i_{\text{dest}}}$, the optic row with the item's destination. Each processor on the optical mesh has a counter that keeps track of the number of items that were moved to it. Every time an item is moved to a processor on the optical mesh, the counter is incremented by one and the item is discarded. The result of this is that each processor on the optical mesh in (A_i, B_j) knows $count(i,j)$, the number of items in B_j that have a destination in A_i.
- A scan operation is performed on the counters on the optical mesh that determines the number of items at processors in columns numbered less than each processor's column number in the same row. A reduction is also performed in each A_i to determine the number of items in that row. With this information, each item is tagged with the column number, j_{int}, of the optical connections it must use in order for items to be distributed as evenly as possible among the s processors on the optical mesh in each submesh A_i. That is, each item is assigned an $n/s \times n/s$ submesh, in $A_{i_{\text{dest}}}$ and $B_{j_{\text{int}}}$, as an intermediate location. Specifically, for an item starting in B_j with a destination in $A_{i_{\text{dest}}}$, if x is the number of items with a destination in the same submesh $A_{i_{\text{dest}}}$ and in a submesh $B_{j'}$, $j' < j$, y is the number of items with a destination in the same submesh $A_{i_{\text{dest}}}$ and z is the index of the item out of those from B_j with destination $A_{i_{\text{dest}}}$, then the item has an intermediate location in column $j_{\text{int}} = \lfloor \frac{x+z}{y} \cdot s \rfloor$ of optics. To tag each item, the items move to the processors on the optical mesh as in the previous step, but instead of being discarded once it reaches the optical mesh processor, it gets tagged and reverses its movement and returns to its original location.
- Items are moved to the diagonal of the mesh, that is, row $i_{\text{int}} = j$ of the optical mesh, which is the intermediate row items move in before moving to their destination row. Then, each item moves to its intermediate column j_{int}, then to its destination row i_{dest}. Once each item is moved to its destination row, it is spread out so that each $n/s \times n/s$ submesh in the row has an equal number of items that have been moved to that row so far.

Now each A_i contains only items that have destinations within A_i. For each A_i, s active processors are used to move s items at a time to their correct destinations.

At each iteration, there are never more than s items moving at a time in each A_i or B_j, so there is enough bandwidth to accomplish each iteration in $O(\sqrt{S})$ time. Since it takes $O(\log N)$

time to reach a processor on the optical mesh and there are N/S iterations, the running time is $\Theta\left(N/\sqrt{S}+\frac{N\log N}{S}\right)=\Theta\left(\frac{1}{\sqrt{r}}\sqrt{N}+\frac{1}{r}\log N\right)$. □

A generalization of the permutation algorithm is needed for the sorting algorithm. We will call it *redistribution*. Instead of a unique destination processor for each item, the mesh is partitioned into approximately square contiguous blocks and each item has a destination block. Within its destination block an item can be assigned to an arbitrary processor as long as each processor ends up with a single item. Within a block, the processor of smallest index is designated as the representative destination location of the block. For simplicity, we assume block sizes are a multiple of N/S. For block sizes less than or equal to N/S, items moving to these blocks use the same data movement as in the permutation algorithm. For blocks of size N/S, there is no change to the permutation algorithm except items designate their destination as the representative processor of their destination block and, in the last step, items just fill in each $n/s \times n/s$ submesh as they arrive by moving the first processor in row-major order of the block without an item yet. For blocks of size greater than N/S, each processor on the $s \times s$ optical mesh has to keep track of whether all the processors in its $n/s \times n/s$ submesh has received an item yet. Before each iteration of the first loop in the permutation algorithm, the S items being moved are moved on the $s \times s$ optical mesh. Each processor on the optical mesh determines how many of the S items will be moved to its submesh, and if it is full, it changes the item's destination to the next available submesh. This takes $\Theta\left(N/\sqrt{S}+\frac{N\log N}{S}\right)=\Theta\left(\frac{1}{\sqrt{r}}\sqrt{N}+\frac{1}{r}\log N\right)$ time.

Theorem 5.2. *On a mesh of size N with an optical pyramid, N items can be sorted in $\Theta\left(\frac{1}{\sqrt{r}}\sqrt{N}+\frac{1}{r}\log N\right)$ time using rN peak power.*

Proof. The following algorithm sorts items into a Hilbert space-filling curve order. All sorting in any submesh using the standard mesh algorithm or a recursive call is in terms of a Hilbert space-filling curve. If another order is desired, one can switch to any other sorted order by a simple permutation. Algorithm 2 gives an outline of the recursive algorithm that sorts N items with S peak power, for $N^{1/4} \le S \le N$.

Algorithm 2 Sorting algorithm for mesh M of size N with peak power S, for $N^{1/4} \le S \le N$.

```
procedure Sort(M, S)
    if Size(M) = S then
        Standard sort M
    else
        Partition M into submeshes M_i of size √(NS)
        for all M_i do                                  ▷ step 1
            Sort(M_i, S)
        end for
        Select every S^th item as splitter
        Standard sort the N/S splitters                 ▷ step 2
        Redistribute into N/S submeshes M'_i            ▷ step 3
        for all M'_i do                                 ▷ step 4
            Sort(M'_i, S)
        end for
    end if
end procedure
```

The base case of the algorithm occurs when the submesh is of size S, when a standard mesh algorithm [19] can sort the $s \times s$ mesh in $O(\sqrt{S})$ time using peak power S. When $S = N$ this is just the standard mesh sorting algorithm that sorts in $\Theta(\sqrt{N})$ time. There are four steps:

Step 1: the mesh is partitioned into $\sqrt{N/S}$ submeshes of size $\sqrt{NS}$, and each submesh is individually sorted one at a time with S peak power in $O(\sqrt{N})$ time, for a total of $O(N/\sqrt{S})$ time.

Step 2: every S^{th} item in each of the submeshes sorted in step 1 is designated as a splitter and moved to the $n/s \times n/s$ optical mesh, where they are sorted using a standard mesh sorting algorithm. Since $S \geq N^{1/4}$, this takes $O(N/\sqrt{S})$ time.

Step 3: the data is partitioned along a Hilbert curve. Each splitter must determine its correct position in the Hilbert curve ordering, and each item must determine which part it belongs in. To do this, S copies of the splitters in parallel are distributed so that each submesh of size N/S has a copy of the splitters. In each submesh, there is one active processor at any given time and the splitters are merged with the items to determine which part each item belongs in and the number of items from that submesh that belong in each part. When $S > N^{1/3}$, the number of splitters, N/S, is less than the size of an individual submesh, $\sqrt{NS}$, so there are extra copies of the splitters that can be disregarded. Each item individually can determine its part in $O(\log N)$ time by just searching the copy of the splitters. Then a reverse movement happens so that the total number of items in each part for the whole mesh is determined. This data is sent to all the items in the mesh so each item knows the location of the part it needs to move to. Then, the redistribution algorithm is used to move each item in its correct part. This takes $\Theta(N/\sqrt{S})$ time.

Step 4: in the worst case, the size of each part is $O(\sqrt{NS})$, but each part of size $O(\sqrt{NS})$ takes $O(\sqrt{N})$ time to sort, so this step is accomplished in $O(N/\sqrt{S})$ time.

In the case where $S < N^{1/4}$, a few modifications to the algorithm must be made. For steps 1 and 2, the sorting algorithm using peak power $\sqrt{N}$ is simulated. Therefore, in step 1, the items are partitioned into $N^{1/4}$ submeshes of size $N^{3/4}$ and each is recursively sorted with S power. For step 2, the simulation of sorting the $\sqrt{N}$ splitters is run on the $s \times s$ optical mesh, that is, the wire $\sqrt{n}/s \times \sqrt{n}/s$ mesh around each processor in the optical mesh that acts as a submesh of size $\sqrt{N}/S$ part of the $\sqrt{N}$ splitters. Since the total energy required to sort $\sqrt{N}$ items on a mesh is $N^{3/4}$, step 2 takes $O(N^{3/4}/S)$ time, which is within the required time. No other changes are required for the remaining steps, where the redistribution algorithm and recursive calls with S peak power are used, which takes $\Theta\left(N/\sqrt{S} + \frac{N \log N}{S}\right) = \Theta\left(\frac{1}{\sqrt{r}}\sqrt{N} + \frac{1}{r}\log N\right)$ time. □

Given this algorithm for sorting, algorithms that also use sorting or routing can have similar time-power tradeoffs. We will only consider $\Omega(\log^8 N)$ peak power because when peak power is close to 1, algorithms are more serial in nature and are less interesting.

6 All-Nearest-Neighbors

Given a set A of points in d-dimensional space, the all-nearest-neighbors problem is to determine, for every point $p \in A$, the closest point in $A \setminus \{p\}$, where distance is measured via an L_p metric, $1 \leq p \leq \infty$. It is well known that this fundamental problem can be solved in $\Theta(N \log N)$ time serially [34].

Theorem 6.1. *Given N or fewer points in d-dimensional space, distributed one per processor on a mesh of size N with an optical pyramid, the all-nearest-neighbors problem can be solved in $\Theta\left(\frac{1}{\sqrt{r}}\sqrt{N}\right)$ time using rN peak power, for $r \in \Omega\left(\frac{\log^8 N}{N}\right)$, where the implied constants depend upon d.*

Proof. We present the algorithm for $d = 2$. The algorithm for higher dimensions is the same, with only the various constants changing (e.g., number of slabs at each step, number of points that need to be broadcast) as functions of d. The algorithm follows the outline of solving all-nearest-neighbors on the mesh in [18].

The points are first partitioned into five disjoint, linearly separable vertical slabs, with each slab containing $N/5$ points. That is, assuming $p_1, \ldots, p_N$ is a list of the N points sorted by x-coordinate (with ties broken by y-coordinate) and x_1, x_2, x_3, x_4 are the x-coordinates of points $p_{N/5}$, $p_{2N/5}$, $p_{3N/5}$, $p_{4N/5}$ the five *slabs* are the following sets of points:

1. $\{p \mid x\text{-coordinate of } p \leq x_1\}$

2. $\{p \mid x_1 < x\text{-coordinate of } p \leq x_2\}$

3. $\{p \mid x_2 < x\text{-coordinate of } p \leq x_3\}$
4. $\{p \mid x_3 < x\text{-coordinate of } p \leq x_4\}$
5. $\{p \mid x\text{-coordinate of } p > x_4\}$

The all-nearest-neighbors problem is solved within each vertical slab. Likewise, the points are divided into five horizontal slabs and the problem is recursively solved in each slab. By a lemma proven in [18], there are at most 8 points in each rectangular region determined by the intersection of a vertical slab and a horizontal slab that has not determined its true closest neighbor and these 8 points can be identified efficiently. A broadcast of these 8 points from each of the 25 rectangular regions is then used to determine the true nearest neighbors of these points.

In order for the problem to be recursively solved in a square submesh for each of the slabs, the points are sorted using Hilbert curve ordering. Points are sorted by x-coordinate for vertical slabs and by y-coordinate for horizontal slabs. Determining the 8 points in each rectangular region that may not know their true nearest neighbor can be accomplished by sorting the points in each region and performing a reduction operation. In our algorithm, the available power is divided evenly among the recursive calls so that recursive calls can be executed in parallel. Therefore, the running time obeys the recurrence $T(N') = T_{\text{sort}}(N') + 2T(\frac{N'}{5})$, where $T_{\text{sort}}(N')$ is the time to sort N' items with $S/\frac{N}{N'}$ power.

We define the base case of our algorithm to occur when the power is $\log^2 N$. At this point, there are $\frac{S}{\log^2 N}$ submeshes running in parallel, each of size $\frac{N \log^2 N}{S}$, with $\log^2 N$ available peak power per submesh. Now, a serial algorithm is simulated to solve the all-nearest-neighbors problem. Since the optical pyramid reduces the communication diameter of the mesh to the logarithm of the size of the mesh, a serial algorithm on an input of size M can be simulated on a mesh of size M with $O(\log M)$ overhead. Thus the base case is $T(\frac{N \log^2 N}{S}) \in O\left(\frac{N \log^2 N}{S} \log^2(\frac{N \log^2 N}{S})\right)$.

Since peak power $S \in \Omega(\log^8 N)$, $T_{\text{sort}}(N') \in \Theta\left(\sqrt{\frac{N'N}{S}}\right)$ and the total running time of the algorithm is $\Theta(N/\sqrt{S}) = \Theta\left(\frac{1}{\sqrt{r}}\sqrt{N}\right)$. □

7 Minimum Spanning Forest

Often parallel algorithms for a graph given as an adjacency matrix are faster than those for when they are given as a set of edges, and this holds true for finding a minimal spanning forest on the mesh with optics. However, for large graphs, a more natural input format is to be given the graph as a set of edges. Here we only give an algorithm for the harder case. For adjacency matrix input it can be shown that a minimal spanning forest can be found in $\Theta\left(\frac{1}{r}\log N\right)$ time, for $r \in O\left(\frac{\log N}{N^{1/4}}\right)$.

Theorem 7.1. *Given N weighted edges of an undirected graph arbitrarily distributed one edge per processor on a mesh of size N with an optical pyramid, a minimum spanning forest can be determined in $\Theta\left(\frac{1}{\sqrt{r}}\sqrt{N}\right)$ time using rN peak power, for $r \in \Omega\left(\frac{\log^8 N}{N}\right)$.*

Proof. For simplicity, each edge is represented twice so that an edge between vertices u and v is stored in one processor as (u, v) and in another as (v, u). Also assume that every vertex has an edge to itself as a way of ensuring it is represented.

The algorithm uses a series of recursive steps, commonly called *Borůvka steps*, where for each vertex an incident edge of smallest weight is selected. The resulting subgraph consists of edges in the minimum spanning forest, and they form trees which are *supervertices*, i.e., vertices for the following stages. For each tree, one of the vertices is chosen and its label becomes the label for the supervertex. Then some of the original edges in the graph become edges between supervertices, where the edge between supervertices U and V is the one having minimal weight among all edges connecting a vertex in U with one in V. This is known as *vertex reduction*.

1. Do vertex reduction five times. The number of vertices is now no more than 1/32 of the original number.

2. In each quadrant of the mesh, recursively solve the problem using only edges in the quadrant. The number of edges selected in each quadrant is proportional to the number of supervertices, so for all the quadrants combined, the number of edges is at most $4 * (1/32) = 1/8$ the number of original vertices.

3. Move these edges to a submesh of size $N/8$, and recursively solve the problem in this submesh. This uses the fact that a minimum spanning forest of the entire graph is a minimum spanning forest of the union of the subgraphs.

Power is divided evenly among the parallel recursive calls. If T_{MSF} is the time to find a minimum spanning forest, T_{VR} is the time to do a vertex reduction and T_{sort} is the time to sort (to move edges to a submesh of size $N/8$), then $T_{\mathrm{MSF}}(N, S) = 5T_{\mathrm{VR}}(N, S) + T_{\mathrm{MSF}}(N/4, S/4) + T_{\mathrm{MSF}}(N/8, S) + T_{\mathrm{sort}}(N, S)$

Vertex reductions are done recursively, using upward tree reductions at each step, which are themselves done recursively. In an upward tree reduction, there is a directed tree with edges pointing toward the root. Each vertex has a value, and the result of the value is a semigroup operation applied to all of these values. See [2, 19, 29] for an explanation of how these operations are used. If T_{UT} is the time for doing upward tree reduction, then

$$T_{\mathrm{VR}}(N, S) = T_{\mathrm{UT}}(N, S) + T_{\mathrm{VR}}(N/2, S) + T_{\mathrm{sort}}(N, S)$$

where

$$T_{\mathrm{UT}}(N, S) = T_{\mathrm{sort}}(N, S) + T_{\mathrm{UT}}(N/4, S/4)$$

Similar to the all-nearest-neighbors algorithm, before the power available to a recursive call becomes too small (less than the square of the logarithm of the size of mesh), a serial algorithm is simulated to solve the lowest levels of recursion. Since a minimum spanning forest can be computed in $O(N \log N)$ time serially, $T_{\mathrm{MSF}}(N, S) \in \Theta(N/\sqrt{S}) = \Theta\left(\frac{1}{\sqrt{r}}\sqrt{N}\right)$. □

Finding the minimum spanning forest of a graph is often a key step in many other graph algorithms. Algorithms for finding connected components, biconnected components, bridge edges, and articulation points follow almost immediately [1, 2, 19].

Corollary 7.2. *The connected components, biconnected components, bridge edges, and articulation points of a graph with N edges can be found in $\Theta\left(\frac{1}{\sqrt{r}}\sqrt{N}\right)$ time using rN peak power, for $r \in \Omega\left(\frac{\log^8 N}{N}\right)$, on a mesh of size N with an optical pyramid.* □

8 Conclusion

Energy and peak power are becoming are increasingly important in parallel computing. E.g., the DOE report *Architectures and Technology for Extreme Scale Computing* [28] states:

> The primary design constraint for future HPC systems will be power consumption. ... Data movement will be a bigger factor for system energy consumption and cost than FLOP/s. ... Energy and performance costs should be reflected in abstract machine model.

Unfortunately few parallel algorithms address the energy consumption problem. It is addressed in some algorithms for sensor networks, but they are limited by the total energy available in their batteries, while parallel computers are limited by peak power which is supplied externally.

Our power aware algorithms address these issues, considering fundamental tradeoffs of time versus peak power for communication intensive problems. Our abstract model is based on ideas first expressed in von Neumann's finite automata model which addressed physical locality and data movement. To this we added a model of on-chip optical connections, a capability which is rapidly becoming available and which offers the possibility of reducing time and/or energy. As the number of

	Time
Routing k items	$O(k + \log N)$
Broadcast	$\Theta\left(\frac{1}{r} + \log N\right)$
Reduction	
Scan	
Permutation	$\Theta\left(\frac{1}{\sqrt{r}}\sqrt{N} + \frac{1}{r}\log N\right)$
Sorting	
All-Nearest-Neighbors	$\Theta\left(\frac{1}{\sqrt{r}}\sqrt{N}\right)$, $r \in \Omega\left(\frac{\log^8 N}{N}\right)$
Minimum Spanning Forest (Edge List)	
Minimum Spanning Forest (Adjacency Matrix)	$\Theta\left(\frac{1}{r}\log N + N^{1/4}\right)$

Table 1: Summary of running times with N processors and r peak power.

processors greatly increases, the asymptotic bounds of our algorithms are descriptive of the behavior of their running times.

The optical interconnects form a pyramid which we use for problems quite unlike its previous roles in parallel computing. It is a fundamental layout appearing in VLSI design as well as being a model of parallelism studied for problems involving images, adjacency matrices, etc. [3, 7, 8, 19, 33, 39]. However, simple bandwidth arguments show that it cannot sort faster than the base mesh; therefore, the pyramid has no advantages for problems which require sorting. This suddenly changes when peak power is limited, though new approaches are needed.

Table 1 summarizes the running times given in this paper. In addition, we showed that the optical pyramid layout is optimal in terms of communication diameter and time-power tradeoff for sorting. Using the pyramid, we achieved a non-linear time/peak energy tradeoff, where if the peak power is cut in half then the time increases by only a factor of $\sqrt{2}$, instead of the factor of 2 that occurs with stepwise simulation. Similar results were obtained for problems where the input was an unstructured set of edges or points. The algorithms presented combine parallel divide-and-conquer approaches with stepwise simulation of serial algorithms when there is only one active processor per submesh.

These results explore a new perspective for modeling energy usage on massively parallel architectures and emerging capabilities. The actual implementations of these algorithms and realization of the model in hardware is another area of research, but it is abstract enough to be applied to moderately different computer architectures. For example, depending on the physical properties of the interconnection technology, it may be the case that the communication over some of the shorter optics in our model are more efficiently implemented using standard electrical wires. Nevertheless, the basic principles of routing data with numerous processors and power constraints shown in this work still hold. Further, they hold for any technology which can supply a layer of interconnections which can transmit information long distances with low power relative to standard wire interconnections.

Note that one energy-reducing hardware option, reducing the clock as the voltage is decreased, can be utilized in conjunction with our algorithms. Since the algorithms almost always have $S = rN$ processors active at any one time, one merely needs to introduce a multiplicative factor for a tradeoff of increasing peak power versus decreasing clock speed.

The continued study of the interplay between time and energy usage of algorithms on parallel computers is necessary for the future. In [30] it is shown that, for some problems, peak power usage can be reduced, without increasing the time, on the standard mesh without optics. Depending on advancements of computer architecture and fabrication technology, we will continue to need the development of theory and models of computation. Extensions include analyzing algorithms on models with more than one layer of optics or 3-dimensional meshes. It can be shown that for 2-dimensional meshes, two layers of noncrossing optics is asymptotically as powerful as any constant number of layers of noncrossing optics [23]. In 3-dimensional meshes the problem of optical pathways crossing is eliminated, which allows for more optical connections and bandwidth on them. Further, the underlying 3D mesh has a smaller diameter and larger bisection bandwidth than the 2D mesh.

Acknowledgements

Research partially supported by NSF grant CDI-1027192, DOE grant DE-FC52-08NA28616 and King Abdulaziz University.

References

[1] M. J. Atallah and S. E. Hambrusch. Solving tree problems on a mesh-connected processor array. *Information and Control*, 69(1-3):168–187, 1986.

[2] M. J. Atallah and S. R. Kosaraju. Graph problems on a mesh-connected processor array. *Journal of the ACM*, 31(3):649–667, July 1984.

[3] C. Cantoni and S. Levialdi, editors. *Pyramidal Systems for Computer Vision*, volume 25 of *NATO ASI Series F: Computer and Systems Sciences*. Springer-Verlag, 1986.

[4] J. Chan, G. Hendry, A. Biberman, and K. Bergman. Architectural exploration of chip-scale photonic interconnection network designs using physical-layer analysis. *IEEE/OSA Journal of Lightwave Technology*, 28(9):1305–1315, May 2010.

[5] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ISCA '11, pages 365–376, New York, NY, USA, 2011. ACM.

[6] M. D. Grammatikakis, D. F. Hsu, M. Kraetzl, and J. F. Sibeyn. Packet routing in fixed-connection networks: A survey. *Journal of Parallel and Distributed Computing*, 54(2):77–132, 1998.

[7] R. I. Greenberg. The fat-pyramid and universal parallel computation independent of wire delay. *IEEE Transactions on Computers*, 43(12):1358–1364, 1994.

[8] G. E. Jan, S.-W. Leu, and C.-H. Li. On the array embeddings and layouts of quadtrees and pyramids. *Journal of Information Science and Engineering*, 20(1):127–141, 2004.

[9] A. Joshi, C. Batten, Y.-J. Kwon, S. Beamer, I. Shamim, K. Asanovic, and V. Stojanovic. Silicon-photonic clos networks for global on-chip communication. In *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '09, pages 124–133, Washington, DC, USA, 2009. IEEE Computer Society.

[10] D. J. Kleitman. The crossing number of $K_{5,n}$. *Journal of Combinatorial Theory*, 9(4):315–323, 1970.

[11] P. Kogge, K. Bergman, S. Borkar, D. Campbell, W. Carson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, and Others. ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems. Technical report, University of Notre Dame, CSE Dept., 2008.

[12] M. Kumar and D. S. Hirschberg. An efficient implementation of Batcher's odd-even merge algorithm and its application in parallel sorting schemes. *IEEE Transactions on Computers*, 32:254–264, 1983.

[13] G. Kurian, J. E. Miller, J. Psota, J. Eastep, J. Liu, J. Michel, L. C. Kimerling, and A. Agarwal. ATAC: a 1000-core cache-coherent processor with on-chip optical network. In *Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques*, PACT '10, pages 477–488, New York, NY, USA, 2010. ACM.

[14] F. T. Leighton. *Complexity Issues in VLSI.* The MIT Press, 1983.

[15] J. Liu, X. Sun, R. Camacho-Aguilera, L. C. Kimerling, and J. Michel. Ge-on-Si laser operating at room temperature. *Opt. Lett.*, 35(5):679–681, 2010.

[16] Y. Ma, S. Sen, and I. D. Scherson. The distance bound for sorting on mesh-connected processor arrays is tight. In *27th Annual Symposium on Foundations of Computer Science*, pages 255–263, Oct. 1986.

[17] J. D. Meindl, J. A. Davis, P. Zarkesh-Ha, C. S. Patel, K. P. Martin, and P. A. Kohl. Interconnect opportunities for gigascale integration. *IBM Journal of Research and Development*, 46(2.3):245–263, Mar. 2002.

[18] R. Miller and Q. F. Stout. Mesh computer algorithms for computational geometry. *IEEE Transactions on Computers*, 38(3):321–340, Mar. 1989.

[19] R. Miller and Q. F. Stout. *Parallel Algorithms for Regular Architectures: Meshes and Pyramids.* The MIT Press, 1996.

[20] D. Nassimi and S. Sahni. Bitonic sort on a mesh-connected parallel computer. *IEEE Transactions on Computers*, 28:2–7, 1979.

[21] I. O'Connor and F. Gaffiot. On-chip optical interconnect for low-power. In E. Macii, editor, *Ultra Low-Power Electronics and Design*, pages 21–39. Springer US, 2004.

[22] K. Ohashi, K. Nishi, T. Shimizu, M. Nakada, J. Fujikata, J. Ushida, S. Torii, K. Nose, M. Mizuno, H. Yukawa, M. Kinoshita, N. Suzuki, A. Gomyo, T. Ishi, D. Okamoto, K. Furue, T. Ueno, T. Tsuchizawa, T. Watanabe, K. Yamada, S.-i. Itabashi, and J. Akedo. On-chip optical interconnect. *Proceedings of the IEEE*, 97(7):1186–1198, July 2009.

[23] P. Poon. *Energy-Efficient Algorithms on Mesh-Connected Systems with Additional Communication Links.* PhD thesis, University of Michigan, 2013.

[24] I. D. Scherson and S. Sen. Parallel sorting in two-dimensional VLSI models of computation. *IEEE Transactions on Computers*, 38(2):238–249, Feb. 1989.

[25] I. D. Scherson, S. Sen, and Y. Ma. Two nearly optimal sorting algorithms for mesh-connected processor arrays using shear-sort. *Journal of Parallel and Distributed Computing*, 6(1):151–165, 1989.

[26] C. P. Schnorr and A. Shamir. An optimal sorting algorithm for mesh connected computers. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 255–263, New York, NY, USA, 1986. ACM.

[27] J. Shalf, S. Dosanjh, and J. Morrison. Exascale computing technology challenges. In J. Palma, M. Daydé, O. Marques, and J. a. Lopes, editors, *High Performance Computing for Computational Science—VECPAR 2010*, volume 6449 of *Lecture Notes in Computer Science*, pages 1–25. Springer Berlin / Heidelberg, 2011.

[28] R. Stevens, A. White, and et al. Scientific grand challenges: Architectures and technology for extreme scale computing, 2009.

[29] Q. F. Stout. Tree-based graph algorithms for some parallel computers. In *Proceedings of the 1985 International Conference on Parallel Processing*, pages 727–730. CRC Press, 1985.

[30] Q. F. Stout. Minimizing peak energy on mesh-connected systems. In *Proceedings of the Eighteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '06, pages 331–331, New York, NY, USA, 2006. ACM.

[31] C. D. Thompson and H. T. Kung. Sorting on a mesh-connected parallel computer. *Communications of the ACM*, 2:263–271, Apr. 1977.

[32] T. Toffoli and N. Margolus. *Cellular Automata Machines: A New Environment for Modeling.* MIT Press, 1987.

[33] L. Uhr, editor. *Parallel Computer Vision*. Academic Press, 1987.

[34] P. Vaidya. An $O(n \log n)$ algorithm for the all-nearest-neighbors problem. *Discrete & Computational Geometry*, 4:101–115, 1989.

[35] G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, J. Lugo-Martinez, S. Swanson, and M. B. Taylor. Conservation cores: reducing the energy of mature computations. In *Proceedings of the Fifteenth Edition on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '10, pages 205–218, New York, NY, USA, 2010. ACM.

[36] Y. A. Vlasov. Silicon photonics for next generation computing systems. In *34th European Conference on Optical Communication, 2008, ECOC 2008*, pages 1–2, Sept. 2008.

[37] J. von Neumann and A. W. Burks. *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, IL, USA, 1966.

[38] S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

[39] T. Yamada, N. Fujii, and S. Ueno. On three-dimensional layout of pyramid networks. In *APCCAS*, pages 159–164, 2002.